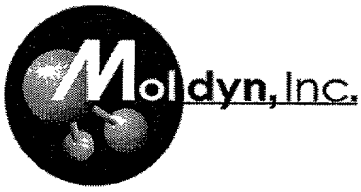
APPENDIX A
SBIL-119 63135-011

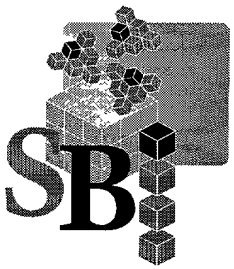
Optimal Integrators for Reduced Variables Molecular Dynamics

Final Report
Air Force SBIR Phase II
Contract No. F49620-96-C-0035

Submitted by

Dr. Carlos E. Padilla
Dr. Valeri I. Karlov
Dr. Glenn E. Webb
Dr. Eric P. Koistinen
Dr. Dmitri Beglov
Dr. Hon M. Chun





SBI-Moldyn, Inc.
955 Massachusetts Avenue
Cambridge, MA 02139
Phone (617) 354-3124

Fax (617) 491-4522
www.moldyn.com

Acknowledgments

SBI-Moldyn Inc. gratefully acknowledges the support of the AFOSR SBIR Program (through contract number F49620-96-C-0035) in carrying out this important and significant research work under topic AF96-002, Software for Computational Chemistry. This support made it possible to develop a new powerful MBO(N)D-II code for large-scale simulations of reduced-variable molecular dynamics. The role of molecular dynamics in biotechnology has significantly grown during the last one-two years due to the need of highly accurate molecular modeling in the fields of structural genomics and proteomics, which have emerged as a result of sequencing the human Genome. Given the latter fact, this AFOSR SBIR Phase II grant was especially timed to enable the creation of the MBO(N)D-II code (a successor of the MBO(N)D code) to meet the challenges of molecular modeling in the post-genomics era.

The major contributors to this effort at Moldyn were Dr. Carlos E. Padilla, who led this effort as Principal Investigator, Dr. Valeri I. Karlov, Dr. Glenn E. Webb, Dr. Eric P. Koistinen, Dr. Dmitri Beglov, and Dr. Hon M. Chun. The authors wish to acknowledge the role of our consultant Prof. John Maddocks who provided significant inputs relating to impetus striction, symplectic integrators, and Euler parameter formulations.

List of Acronyms

ATP	Advanced Technology Program
BCH	Baker-Campbell-Hausdorff
BDF	Backwards Differentiation Formulas
CHARMm	Chemistry at HARvard Macromolecular Mechanics
COM	Center of Mass
CPU	Control Processing Unit
CTF	C-Terminal Fragment
DAE	Differential Algebraic Equations
DISCOS	Dynamics Interaction Simulation of Controls and Structures
DOF	Degree of Freedom
FF	Force-Field
GSN	Gauss-Seidel-Newton
IVP	Initial Value Problem
IVV	Iterated Velocity Verlet
LT	Liouville-Trotter
MBD	Multi-Body Dynamics
MBO(N)D	Multi-Body Order (N) Dynamics
MD	Molecular Dynamics
MM	Molecular Mechanics
MSI	Molecular Simulations, Inc.
MTS	Multiple Time Scale
MTSL	Multiple Time Scale Lobatto
N-DISCOS	Order (N) - Dynamics Interaction Simulation of Controls and Structures
NEIMO	Newton-Euler Inverse Mass Operator
NIST	National Institute of STandards
NSM	Null-Space Method
ODE	Ordinary Differential Equations
OMBI	Optimized Multibody Integrator
OO	Object Oriented
OOP	Object Oriented Programming
P-C	Predictor-Corrector
PRK	Partitioned Runge-Kutta
PV	Position Verlet
RESPA	REference System Propagator Algorithm
RKNX	Runge-Kutta-Nystrom (X is the order of integrator)
RKX	Runge-Kutta (X is the order of integrator)
RSM	Range-Space Method
SBI	Structural Bioinformatics Inc.
SMD	Substructured Molecular Dynamics
SOR	Successive Over-Relaxation
UML	Universal Modeling Language
VCD	Verlet Central Difference
VV	Velocity Verlet

Table of Contents

1 Introduction

This document outlines the work done on the AFOSR SBIR Phase II project. During the Phase II work we have developed the new methods for integrating reduced variables molecular dynamics and finalized their implementation in a prototype of a commercial molecular dynamics (MD) simulation program for biological and polymer applications. The predecessor of this new MD code is the MBO(N)D (MultiBody O(N)D Dynamics) code developed by Moldyn, Inc. and currently distributed by MSI as a multi-body dynamic engine for CHARMm.

In a theoretical part, this work resulted into the development of a new Optimized MultiBody Integrator (OMBI) that dramatically improved MBO(N)D's performance. Specifically, the new OMBI allows for maintaining stability over longer duration simulation timescales, and further improves MBO(N)D's computational speed by at least 2-10 times. But most importantly, we were able to significantly expand the scale of simulation problems to which MBO(N)D may be applied. Namely, now it became possible to run MD simulations for large molecules (substructured into thousands of bodies) with large solvation systems (thousands of explicit water molecules). Correspondingly, our initial plans for Phase II "just to improve" MBO(N)D performance resulted into the development of a principally new capability for large-scale MD simulations in post-genomics era. The new code with OMBI is dubbed as MBO(N)D-II. MBO(N)D-II is still a prototype code, and additional developments (on the side of forcefield computations) are needed in order to make it the most advanced commercial software for MD simulations. However, these future developments are beyond the scope of this AFOSR project. But, even as a prototype, MBO(N)D-II offers a great potential to eliminate (or at least significantly reduce) the timestep bottleneck in MD, and begin to achieve the utility for the pharmaceutical and materials industries.

Our only initial strategy for the dissemination of MBO(N)D-II was, as it was with MBO(N)D, to incorporate it into the MSI family for commercial sales, and into the Academic CHARMm for distribution to the educational institutions. However, given a new important fact that during the Phase II work Moldyn, Inc. was acquired by Structural Bioinformatics Inc. (SBI, http://www.strubix.com), MBO(N)D-II is planned to be incorporated into SBI's product lines for structure-based drug design. Also, due to the fact that MBO(N)D-II uses principally new multibody dynamics SBI might distribute it as a new commercial code to fill the need for faster and more accurate molecular dynamics in the fields of genomics and proteomics.

2 Significance

2.1 Multibody Dynamics for Substructured MD

Over the past decade, numerous classical all-atom molecular dynamics (MD) simulation methods have been developed with the promise of revolutionizing chemical design-intensive industries by enabling computer-based research, design and analysis of large molecular systems (e.g., proteins, polymers) prior to laboratory synthesis and testing. Effectively, these MD methods model the time-evolving classical dynamics trajectory of each individual atom in a molecule, from which important physical properties can be derived.

Indeed, the pharmaceutical industry, particularly, embraced the promise of MD early-on and has spent much of the past decade attempting to use it to support drug design research. Completion of sequencing the human Genome by both government and private organizations

(National Human genome research Institute lead by Francis S. Collins and Celera Genomics lead by Craig Venter) have significantly boosted the role of molecular dynamics in biotechnology. Now when the sequences of hundreds of thousands of proteins are available, and 3D protein structures for them are evaluated via homology methods, there is a need in highly accurate molecular modeling via MD simulations. In a fact, our new parent company Structural Bioinformatics Inc. (SBI) is now making MD simulations as a part of high-throughput process for creating 4D structural databases for Genome sequences. Note that the "fourth" dimension here entails "time", i.e. the 4D database will also provide dynamic information about 3D protein structures. This dynamic protein structural information is used in the generation of DynaPharm™ templates – virtual constructs of protein pharmacophores playing an important role in structure-based drug design.

Beyond the biotechnology applications, the materials industry has explored the use of MD (but to a somewhat lesser extent) in designing polymers, composites, and other new materials.

Unfortunately, the utility of classical all-atom MD has not met expectations, primarily because it is restricted to severely short timesteps (0.5 - 1 femtoseconds) required to handle the high frequency content in the dynamics equations. In application to large biological and polymeric molecules, where the event durations of interest are in the nano-, micro-, and in some cases milli-second domains, such short timesteps result in a computationally intensive process. Given that drug and material design researchers typically examine tens-to-hundreds of thousands of candidate leads and derivatives in the search for a new product, the use of all-atom MD becomes prohibitive, and hence its utility has been severely limited.

Over the past seven years, Moldyn has been at the forefront of research efforts to overcome this short timestep bottleneck. The highly promising approach that has emerged, known as MBO(N)D (Multibody Order (N) Dynamics), aggregates, or substructures, groups of atoms in a molecule into flexible (or rigid) bodies in order to simulate essential low frequency dynamics. Elimination of the unimportant high frequency content through this multibody approach allows the use of much longer timesteps.

It has become evident that the integrator utilized for multibody dynamics is critical to successfully attaining long timesteps for SMD, particularly with respect to stability, and the number of forcefield evaluations required for each step. Recognizing this early-on, we applied for and were awarded the Phase I and Phase II projects, which have successfully demonstrated the superiority of a new class of integrators as well as integrator/formulation combinations. These developments resulted into the development of the MBO(N)D-II code as a successor of the original MBO(N)D (since now on called MBO(N)D-I in the cases when it is needed for clarification).

In the remainder of this section, a brief history of the development of integrators for substructured molecular dynamics (SMD) is provided in order to identify both the problem they are meant to address, as well as the opportunity this presents for Moldyn's MBO(N)D technology. First, the properties of integrators for classical MD are described, together with the limitations they impose. Next, the use of multibody dynamics formulations technology to extend the capabilities of MD through body-based substructuring is presented. Finally, an overview of the OMBI is presented, describing how it addresses the challenging requirements for numerical integration in the MBO(N)D substructured molecular dynamics formalism.

2.2 Integrators for MD

It is non-trivial to analyze the effectiveness of candidate integration algorithms for large-scale MD problems. The classical MD of macromolecules (e.g., proteins, nucleic acids, and polymers) is governed by Newton's equations of dynamics:

$$M\frac{d^2}{dt^2}q = -\nabla_q V(q) \tag{1}$$

where q is the vector of the Cartesian positions of each atom, M is a diagonal mass matrix containing the mass of each atom, and V is the potential energy function. Equation (1) is highly nonlinear and, for typical macromolecules, has a phase space with large dimensionality. In effect, the trajectories prescribed by Eq. (1) exhibit chaotic behavior in the form of high sensitivity to initial conditions. This, coupled to the long time scales necessary for MD, make it impossible to use standard measures such as stability and accuracy (as normally defined in the literature of numerical integrators, i.e., with respect to a given trajectory) to measure the effectiveness of a numerical discretization solution of Eq. (1). This difficulty has led researchers to seek alternative ways to judge the quality of the numerical solutions of Eq. (1). One such alternative consists of showing that the resulting solution trajectory is "close" to the true trajectory of a "nearby system of differential equations" (Ref. 51). This nearby system should possess similar dynamic properties to the original one.

This approach has contributed to the increasing interest in symplectic integrators for use in classical *MD*. The system of Eq. (1) is a Hamiltonian system. Hamiltonian systems possess symplectic invariants. In two dimensions, this means that the flow of the Hamiltonian system preserves areas in phase space. In higher dimensions, this is a much stronger geometrical property of the flow of the system that has as one of its consequences the preservation of volume in the higher dimensional phase space. The relevant result is that it has been shown that a symplectic numerical discretization of a Hamiltonian system will result in trajectories that are the true trajectories of a "nearby" Hamiltonian system. Thus, a numerical integration algorithm that preserves the symplectic property of Eq. (1) is seen as desirable. As it turns out, the consequences of symplecticness for MD are not clear. As a practical matter, however, some very effective integration schemes, such as leapfrog Verlet (Refs. 48, 53), have been shown to be symplectic. Another strong property of the flow (or set of solutions) of a Hamiltonian system is that it is time reversible. Thus numerical integration schemes that exhibit time-reversibility (and have already passed muster with respect to classical stability and accuracy in simpler systems) are also desirable. This property may be of importance for long-term dynamics (Ref. 53). Leapfrog Verlet is also a time reversible discretization scheme.

Thus, using the above measures as indicators of the potential effectiveness of a candidate integration algorithm, researchers have searched for ways to alleviate the restriction on the integration timestep imposed by the high frequency dynamics of Eq. (1). The numerical integration schemes that have been explored are classified in Ref. 51 into four categories. Only three are of interest in the present discussion, and these are:

1) explicit methods that use small timesteps to accurately resolve the high frequencies;

2) implicit methods that inaccurately resolve the high frequencies in order to achieve larger timesteps; and,

3) methods that incorporate constraints to eliminate the high frequencies (e.g., SHAKE and RATTLE (Refs. 9,34,53).

Among the integrators in category 1 are the already mentioned leapfrog Verlet, together with its close relatives Verlet, Velocity Verlet, and position Verlet. In addition, multistep Gear

methods (Ref. 53) and some higher order symmetric multistep methods (Ref. 55) have been tried. These all suffer from the small timestep problem. Among the multistep methods, the Gear methods are not symplectic or time-reversible, and exhibit poor long term energy conservation. The higher order symmetric methods have not been shown to be symplectic but are time-reversible and exhibit good accuracy with minimal forcefield evaluations. It is not believed that significant improvements in integrator efficiency will result, though, for the accuracies needed for MD (Ref. 51).

The integrators in category 2 suffer from two related drawbacks. The first and more obvious drawback is the fact that an implicit integrator, such as the implicit midpoint method (Ref. 56), requires iterative solution. Due to the high cost of evaluating the potential in Eq. (1) (cost is $O(n^2)$ where n is the number of atoms), even when non-bond pairlist cutoffs are used, an iterative solution carries with it a large computational cost. (By contrast, the Verlet family of integrators require only one forcefield evaluation per step.) To overcome this drawback of the implicit integrators, larger timesteps must be taken, which is in line with the original intent in going to category 2 integrators in the first place. However, the second, more subtle, drawback of these implicit integrators appears when larger timesteps are taken. Larger timesteps mean that higher frequency dynamics of the MD system of Eq. (1) are not accurately resolved. While this may be acceptable for linear systems if we are only interested in the low frequency behavior, coupling of principal or "normal" modes of the motion in highly nonlinear systems such as that of Eq. (1) will result in very inaccurate dynamics, even for the low frequencies, if the high frequencies are poorly resolved (c.f. Refs. 51).

Category 3 integrators are the most promising in terms of reducing the high frequency content of the classical MD equations of motion, while reproducing accurate dynamics, if it can be done at a reasonable cost (only one forcefield evaluation per step). Verlet integrators with SHAKE and RATTLE have been shown to be second-order, time-reversible, symplectic discretizations of the system of Eq. (1) modified to incorporate bond length (and angle) constraints. They require only one forcefield evaluation per step, and the iterative solution of the position constraint equations can be made to converge in very few cycles under typical conditions. Unfortunately, the use of SHAKE (or RATTLE) to enforce bond length constraints results in a timestep increase of only a factor of four at best (2 femtosecond (fs) timesteps instead of 0.5-1fs normally required for Eq. 1 in the case of proteins). This increase in timestep is a marked improvement but not nearly enough to allow classical MD to approach the timescales of interest (micro to milliseconds).

2.3 Multibody Dynamics for Substructured MD

Over the past seven years, scientists at Moldyn have been researching the application of multibody dynamics (MBD) modeling techniques to the problem of molecular dynamics of biological and materials macromolecules. The ultimate objective of these efforts is to achieve a reduced-variable MD modeling capability that will reduce the time required for MD by orders of magnitude and thus enable long-time simulations into the timescales of interest to the scientific and commercial drug design and materials communities. This is to be achieved by a combination of techniques that include fast forcefield evaluation through multipole approximations, fast MBD algorithms enabled by substructuring, and reduction of high frequency content via the use of substructuring and MBD that will allow the use of very large timesteps. It is this last key element in Moldyn's strategy for fast MD that was the subject of this Phase II work.

MBD techniques as applied to MD work on the principle of reducing high frequency dynamics content through the aggregation of groups of atoms that exhibit correlated motions

into rigid or flexible bodies (the process of substructuring). If rigid bodies are used, this is akin to enforcing "hard" constraints between all the atoms that comprise that body. If the body is flexible, "soft" constraints are enforced which are in spirit similar to those presented in Ref. 51, except that MBD flexible "constraints" are more abstract and are closely tied to the concept of the flexible body. The bodies are connected together in a topological chain (which in all generality also includes closed loops) that usually follows the natural topology of the macromolecule (though this is not a limitation). Hinges between bodies are characterized by relative degrees of freedom (DOF) between the bodies, and these DOF can be free, fixed (constrained), or their motion can be prescribed (rheonomic constraints). From the aforegoing discussion, it is evident that these MBD techniques properly fall into the category 3 of Ref. 51.

Moldyn has been at the forefront of the efforts to apply MBD algorithms to MD. Moldyn's MBO(N)D code is the leading example of reduced-variables (flexible multibody) molecular dynamics software. MBO(N)D, which has its roots in spacecraft dynamics, can handle arbitrary multigranular combinations of rigid and flexible atomic substructures (or bodies), together with atomistic and quantum regions, to model a macromolecule. MBO(N)D has been interfaced to academic CHARMm, X-PLOR, and AMBER. These codes provide the molecular mechanics (MM) forcefields to drive the MBO(N)D substructured dynamics representation of the macromolecule.

As is the case for classical MD, the quality of the numerical solutions of MBD equations depends on the properties of the numerical integrators. In fact, it has been pointed out extensively in the MBD literature (Ref. 52) that formulation and numerical integration are intimately related for MBD systems. It should come as no surprise, then, that integrators that work very well for classical MD systems are not effective for MBD systems. What is perhaps less expected is that classical integrators that have worked well for mechanical and aerospace MBD systems do not perform well when applied to MBD systems for substructured MD (SMD). It was the objective of this SBIR project to identify the best integrator or integrator/formulation combination that will bring to fruition the promise of SMD as a method to allow the largest possible timesteps (or faforable combination of multiple timesteps) for large-scale MD simulations. The current MBO(N)D integrator (dubbed Lobatto for short), while better than classical MBD system integrators, still falls short of the qualities of a Verlet algorithm for classical MD. As mentioned at the beginning of this section in the context of classical MD, the selected numerical discretization scheme for the MBD equations of motion should maintain the dynamical properties of the system in some sense.

In the remainder of this section we briefly address the reason for the inadequacy of classical MD (Verlet family) integrators for SMD. We then describe in some detail the issues associated with MBD integration schemes.

2.4 Integrators for SMD

As we have seen above, some category 1 explicit integrators for classical MD (Verlet integrators in particular) have desirable symplectic and time-reversibility properties. These properties translate into excellent long-term stability of energy as well as linear and angular momenta. In addition, they require only one function evaluation per step. What has not been mentioned yet is the fact that these Verlet discretizations take advantage of the particular form of the Hamiltonian for the system of Eq. (1) by "splitting" it (Ref. 51). This process is similar to the Trotter decomposition of the Liouville operator formalism presented in Ref. 45. Due to the simple form of the Hamiltonian, the equations for the derivatives of the momenta do not depend on the momenta themselves (the right hand side or "forcing" terms depend only on the generalized coordinates). This simple form contributes to the simplicity and effectiveness of the

Verlet algorithm. Unfortunately, as we shall see below, MBD equations do not possess the same simple structure. Instead, terms quadratic in the velocity variables appear due to gyroscopic impressed forces. This precludes a straightforward implementation of Verlet integrators to SMD equations.

The field of multibody dynamics has its roots in mechanical and aerospace applications and has been extensively developed over the last twenty years; with the bulk of the developments for spatial chains of flexible bodies and recursive formulations for flexible robotic manipulators coming in the last ten years (see for example Refs. 11, 29, 25, 5, 60, 61). Driving the technology were aerospace applications concerned with large flexible spacecraft with multiple "bodies", as well as light, flexible, robotic manipulators for space applications. Multibody systems are characterized by b bodies (with $n = 6*b$ DOF), which can be rigid or flexible (in which case $n = 6*b+f$, where f is the total number of flexible degrees of freedom), connected together with a certain topology "enforced" by m constraints between pairs of bodies. The generic equations of motion for such a system, which can be derived by a variety of formalisms, is given by (Ref. 52):

$$\left(\begin{array}{c|c} M(q) & [g'(q)]^T \\ \hline g'(q) & 0 \end{array}\right)\left(\begin{array}{c} \ddot{q} \\ \hline \lambda \end{array}\right)=\left(\begin{array}{c} G(t,q,\dot{q}) \\ \hline \gamma(q,\dot{q}) \end{array}\right) \tag{2}$$

together with

$$g(q)=0 \tag{3}$$

$$g_v(q,\dot{q})=g'(q)\dot{q}=0 \quad (=\dot{g}) \tag{4}$$

where q is the n x 1 position vector, $M(q)$ is the mass matrix, $G(t,q,dq/dt)$ is the vector of generalized applied and impressed forces, λ is the L x 1 vector of Lagrange multipliers, $g(q)$ is the L x 1 vector of algebraic constraints enforcing the constraints between bodies, and $g'(q)$ is the L x n constraint matrix (gradient of $g(q)$).

Equations (2)-(4) represent a set of differential-algebraic equations (DAE's) for the trajectories of the MBD system. If only Eq. (2) needed to be satisfied, it would be a system of type 1 which can be solved via direct integration of an initial value problem (IVP), given consistent initial conditions, upon inversion of the augmented mass matrix in the left-hand side of Eq. (2). If both Eqs. (2) and (4) needed to be satisfied, then the system would be of type 2. Finally, if Eqs. (2) and (3) need to be satisfied, the system is of type 3. Note that Eq. (2) incorporates the second time derivative of the position level algebraic constraints of Eq. (3). The numerous existing MBD equation formulations differ only in the specific nature of the generalized position vector q and in the manner in which the DAE of Eqs (2)-(4) is solved. In all cases of spatial systems, the vector $G(t,q,dq/dt)$ of generalized, applied and impressed forces contains quadratic dependencies on dq/dt.

If the system of bodies is connected together such that there is a main chain with "branches" but no closed loops, it is said to have tree topology. Otherwise, it is said to have closed loops. If the system has tree topology, it is possible to express it in *minimal form* by explicitly eliminating all hinge constraints and choosing minimal coordinates. To transform Eqs. (2)-(4) into *minimal form* ignore the lower block (λ-block) of the block vector Eq. (2), as well as Eqs. (3) and (4). In this case, the problem reduces to that of solving a minimal DOF, linearly-implicit, second-order ordinary differential equation (ODE) given initial conditions. Since the mass matrix is positive definite, it is straightforward solve for the accelerations and then to convert the second-order ODE into a set of first-order ODE's, obtaining the classical initial value problem (IVP). If the system has closed loops, it is no longer possible to simplify the structure of

Eqs. (2)-(4) and the *descriptor* or *full descriptor forms* both can be represented by these equations. The system is said to have *descriptor* form if some hinges are modeled by explicit algebraic constraints (e.g., the *cut* joints in closed loop systems), while the rest are modeled by minimal coordinates after explicit elimination of the constraints. The system has *full descriptor form* when all hinges are modeled by algebraic constraints (Eq. (3)).

For systems in *descriptor* or *full descriptor* form, the solution of the DAE is conceptually simple: first, solve for the accelerations and Lagrange multipliers by inverting the matrix on the left-hand side of Eq. (2); second, solve the associated IVP after converting the equations for the acceleration into first-order ODE form. If the system is of type 1 we are done, in principle. If the system is of type 3, solution of Eq. (2) is not sufficient to guarantee satisfaction of the constraint Eq. (3) due to numerical integration drift. Equation (3) can be enforced in a variety of ways ranging from projections of the solution of Eq. (2) into the constraint mainfold via nonlinear iterative solutions starting from a first guess usually given by the solutions of Eq. (2) (Ref. 52) to constraint stabilization of the type 1 equation solution (Refs. 11, 12). The more effective solutions require iterative solutions of Eq. (3).

Thus, the problem of solving the generic DAE of Eqs. (2)-(4) can be seen to be composed of three distinct stages: 1) solution of Eq. (2) for the acceleration and the Lagrange multipliers; 2) IVP integration; and, 3) iterative solution of Eq. (3) to enforce the constraints. As mentioned earlier, effective numerical solutions of the MBD equations of motion will account for the interrelationships between all three stages. The last two stages are the same as for classical MD, with the notable difference that this form of the IVP equations is fundamentally different.

Let us consider stage 1. The block matrix on the left-hand side of Eq. (2) can be shown to be non-singular if the constraints are non-redundant. Solution of the linear implicit method can be carried out directly on Eq. (2) using RSM (Ref. 52) or NSM (Ref. 52) with cost $O(n+m)^3$. However, this matrix is very sparse and has a particular form that has been exploited to yield $O(n+m)$ solutions (for tree topology, with $O(n_l^3)$ for loops). MBO(N)D makes full use of these latest formulations to yield an $O(n+m)$ algorithm for the solution of the dynamics. It should be pointed out that neither the $O(n+m)$, nor the $O(n+m)^3$ solutions require us to select a particular set of generalized coordinates and velocities (cf. Refs. 11, 12). This choice, however, will affect the details of the particular formulation and, most importantly, will have a significant influence on the effectiveness of algorithms for stages 2 and 3.

A variety of well know integrators are used in the MBD field to solve stage 2. These range from the "workhorse" RK4 (Ref. 6) to implicit midpoint and other Backwards Differentiation Formulas (BDF) (Ref. 6) to Adams-Bashforth-Moulton and other Predictor-Corrector (P-C) methods. These integrators have proved adequate for typical MBD application where a relatively small number of DOF is integrated for a short time and high accuracy is the goal. High order integrators with multiple function evaluations are adequate. Symplecticness and time-reversibility have not been an issue. This clearly changes when MBD is applied to SMD. Now a minimal number of FF evaluations (namely, one per step), is required, while long-term stability and "closeness" to a nearby dynamical system is more important than accuracy to a given trajectory afforded by the higher orders. RK4 and P-C methods require four and two FF evaluations, respectively, while exhibiting poor long term stability. BDF methods are also costly due to the iterative solutions, which require multiple FF evaluations, and have been shown to be inferior to Verlet with SHAKE for classical MD applications (Ref. 51).

The recent interest in symplectic integrators has lead applied mathematicians to formulate such integrators for rigid body systems (Ref. 58). While promising, these integrators require a particulation artifact that is not readily extensible to flexible bodies. Other attempts at

adapting classical MD integrators to MBD treatment of SMD rely on modifying Verlet-type integrators to account for the velocity dependent terms (Ref.59). Moldyn has been at the forefront of these efforts (see the rest of this Report). Essentially, an inexact "splitting" between position and velocity variables is attempted in order to use the Verlet structure, and the velocity equations are iterated (see IVV, see VCD below.) Straightforward implementation of these iterative solutions does not result in an optimal implementation. It is necessary to carefully design an integrator that will retain the important properties of time-reversibility and (although harder to prove) symplecticness. Researchers at Moldyn have developed such an integrator (Lobatto integrator for multibody dynamics) under the NIST ATP project.

It was mentioned earlier that the need for iterative solutions of the constraint equations (stage 3 for the MBD DAE solution) is common to both classical MD and MBD systems. Furthermore, similar techniques (eq. Newton-Raphson, Gauss-Seidel) can be used in both cases. The type of constraints that concern us for SMD systems are mainly limited to bond length constraints between bodies and atoms. This suggests that the applications of SHAKE-like GS iterations (modeled to include SOR) could be effective given a suitable optimal integrator.

In the above discussion we have sought to demonstrate how SMD is the next breakthrough step in the quest for more efficient MD simulations. The MBD formalism that enables SMD has unique characteristics that preclude straightforward modifications of classical MD integrators from being used; while at the same time, classical MBD integrators are not suitable for the unique requirements of MD. The goal of this SBIR research project was to address the need for an optimal integrator for SMD that will provide good of numerical properties while enabling the large timesteps that are the promise of SMD. Given the nature of the solutions to MBD equations of motion as presented above it was recognized that rather than a single optimal integrator, an integrator/formulation combination might be necessary. As part of the Phase I, we investigated some higher-order, time-reversible, Lobatto-like integrators (developed in-house) as well as modifications to MBO(N)D's Lobatto integrator. These did not yield significant improvements under the Lobatto integrator, leading us to the conclusion that Lobatto is quasi-optimal for the current implementation of MB. Reformulations of the MBO(N)D equations, however, did result in significant improvements in the effectiveness of the integrator. A Lobatto or RKN variant using momentum variables (instead of velocity) resulted in marked improvements in the long-term stability of the integrator. This is not entirely surprising given that using momentum leads to the natural "splitting" of the underlying Hamiltonian, in manner similar to Verlet discretization of classical MD.

As we shall describe in detail below, the combination of this natural "splitting" of the Hamiltonian (using the Liouville formalism, Ref. 45), together with a novel RESPA-like solution of the multibody equations of motion developed at Moldyn as part of the Phase I and enabled by the use of Euler parameters to describe body rotations, yields the most efficient, stable, MBD integrator/formulation solution for SMD systems (i.e. OMBI solution). Having thus succeeded in finding a significantly better integrator formulation for SMD systems, we have demonstrated feasibility during Phase I; and then, during Phase II, we have finalized this formulation and implemented a powerful new version of MBO(N)D (MBO(N)D-II) for SMD. As was stated above and as it goes throughout of this final report, MBO(N)D-II significantly advanced MD capability for industry and academia.

3 Phase II Accomplishments

3.1 Phase II Objectives and Tasks

The objectives of this AFOSR research work were maintained during Phase II in order ensure the development of the highly effective MBO(N)D-II prototype code for simulation of large-scale MD with solvation.

For reference, the objectives are repeated below from the Phase II Proposal:

- Complete the implementation of best integrators into current version of MBO(N)D/CHARMm
- Implement OMBI in the new version of MBO(N)D
- Implement MTS based on OMBI

For ease of reference, the list of tasks is also shown below:

Task 1. Complete the implementation of high-order modified RKN integrators into the current version of MBO(N)D.

Task 2. Complete the formulation of OMBI for the type of chains of rigid/flexible bodies found in SMD.

Task 3. Implement efficient constraint-handling formulation/architecture.

Task 4. Software implementation of OMBI to create powerful new version of MBO(N)D.

Task 5. Modify Existing MTSL strategy to incorporate OMBI.

Task 6. Assemble prototype code and document in final report.

All objectives and tasks of this AFOSR Phase II SBIR work were successfully fulfilled.

3.2 Summary of Phase II Results

In this section, we provide a summary of the Phase II results. Details of the work performed during Phase II are presented in five Sections (from Section 5 to Section 9). These five Sections correspond to the five Tasks listed above. Task 6 entails writing this final report. Additionally, Section 10 presents the simulation results for three molecular systems using MBO(N)D-II – the code which resulted from the accomplishment of Tasks 1-5. The three molecular systems were the following: 1) 1CTF molecule, 2) 1CTF molecule with explicit solvation model; and, 3) Myosin S1 molecule.

We will first summarize the Phase II results stating that these results lead to the development of a principally new MBO(N)D-II code for reduced variables molecular dynamics, which turned out to be much more powerful than its predecessor MBO(N)D-I. MBO(N)D-II includes the following new features:

- Optimal Multi-Body Integrator (OMBI) with quaternions (Euler parameters) for integrating unconstrained dynamics of particles and rigid bodies – Task 2
- Higher-order OMBI – Task 1
- OMBI with bond-length constraints treated via SHAKE (Gauss-Seidel formalism) – Task 3

- Multiple Time Step (MTS) generalization of OMBI – Task 5
- Object-Oriented (OO) Paradigm for further code growth – Task 4

These new features dramatically improved the MBO(N)D dynamics and integration algorithm

- in speed
- in accuracy
- in memory
- in functionality
- in maintainability

Speed

The new MBO(N)D-II code has 100x faster multi-body dynamics than the O(N) algorithm in the old MBO(N)D-I. This results into up to 10x faster MD simulations since a significant part of simulations involves forcefield calculations. The overall speed-up in MD simulations will be much greater than 10 after further optimization of the forcefield calculations (see Section 4).

Accuracy (and accuracy/speed)

The new MBO(N)D-II code makes it possible to use finer substructuring schemes, and thus achieve better accuracy of MD simulations, without dramatic loss of efficiency. The MBO(N)D-II runs are much more stable due to the fact that OMBI uses quaternions (Euler parameters) which do not have singularities while the O(N) recursive algorithm in the old MBO(N)D-I uses Euler angles which do have singularities. Note that switching between twelve possible Euler angle parametrizations leads to instability in long-time-scale MD simulations with many bodies.

During the Phase II work we have shown that the main potential for fast and accurate MD simulations is not in achieving a longer time step but in exploiting the diversity of the existing multiple time scales (MTS) in the most efficient way. Making a longer time step is, of course, desirable if possible, but it is not the main factor. The old MBO(N)D-I code was based on the philosophy of making the base timestep as long as possible since the O(N) dynamics were expensive. With new and fast OMBI dynamics the base timestep can be small (to retain accuracy) but the large amount of forcefield interactions can be sampled at longer MTS steps (to retain speed).

The fast OMBI algorithm also changes a view on the development of a so-called coarse-grain potential for multi-body molecular dynamics. In this potential one would provide a mechanism of computing forces between collection of atoms rather than between individual atoms. This would lead to sacrifice of accuracy since molecules have atomic structures and atomic forcefield potentials are based on the physics of this structure (e.g. bond interactions and electrostatics). The OMBI/MTS concept offers an excellent alternative via developing a spatial-temporal decomposition of forcefield calculations in which close interactions are treated atomistically while the long-range interactions are computed via a "coarser" model (see Section 4). This approach would be the ultimate solution for achieving the desirable speed without sacrifice of accuracy.

Memory

The new MBO(N)D-II code uses ~10 times less memory than the old MBO(N)D-I code does. This 10-fold advantage would be even larger for larger molecular systems. The memory

limitations of the MBO(N)D-I code come from the complex topological structures of the O(N) algorithm. The new OMBI data structures are simple and memory thrifty.

Functionality

Due its dramatically improved speed, accuracy and memory, MBO(N)D-II provides much more functionality in studying important biological processes via MD simulations. First of all, it has become possible to simulate MD for large molecules with chemically acceptable substructuring (i.e. without lumping a few residues into a body). Second, it is now possible to add a large amount of explicit water molecules in MD simulations.

Maintainability

The use of the OO paradigm in the development of the MBO(N)D-II code makes this code reusable and easily maintainable. The latter entails, for example, adding new capabilities and functionalities via C++ classes. Also, the OO architecture of the MBO(N)D-II code opens the possibility for its massive parallelization.

As was mentioned above, Section 5 to Section 9 provides technical details on Tasks 1-5. Below we only single out the most important theoretical results, which made the basis of the MBO(N)D-II code.

The OBBI is the main theoretical result of this AFOSR SBIR work. The OMBI provides a powerful unified formulation/integration framework for multibody molecular dynamics. We originally developed OMBI as a second-order integrator for multibody systems with rigid bodies. We further generalized the OMBI to its higher-order versions as well as added the capabilities such as handling flexible bodies, accounting for constraints via SHAKE-like Gauss-Seidel correction algorithm, and finally incorporated OMBI into MTS strategies. It is important to emphasize that the development of OMBI in this Phase II AFOSR project provides one of the most significant contribution to modeling rotational motion of rigid bodies since XVIII century when Mr. Euler introduced both Euler angles and Euler parameters (quaternions). Indeed, for over 200 years researchers had the choice of integrating rigid body dynamics either via the use of 3 Euler angles (dealing with the issue of degenerated combination of Euler angles) or via the use of 4 Euler parameters (dealing with the norm constraint on the quaternions). We found a very simple and elegant solution to this problem. In a fact, we enabled the use of 4 Euler parameters with automatical satisfaction of the norm constraint during integration. In other words, all advantages of the Euler parameters (in terms of accuracy and stability) over Euler angles were maintained but their disadvantage due to the need of maintaining the norm constraint was fully eliminated. We solved this problem by realizing that one can optimally tailor the numerical integrator to multi-body dynamics rather than develop model-based integrators (like Runge-Kutta or Adams integrators).

4 Phase III Plan

Encouraged by a high performance (in speed, accuracy, and memory) of the new OMBI dynamics engine in MBO(N)D-II (developed during this AFOSR Phase II work), we are now turning our attention to the forcefield issues. Although these issues were not in the original Phase II plans, it became clear to us that the future most efficient code for large-scale MD simulations need to effectively handle the forcefield calculations, especially those which are associated with large amount of explicit solvent. We have determined that the OMBI/MTS technology developed in this Phase II work can be generalized for optimizing the forcefield calculations via their spatial-temporal decomposition. During the Phase II work we made preliminary investigation to assess the potential of this new unified and powerful approach to speeding-up MD simulations of large solvated systems.

Below we justify and highlight the major points of the Phase III plan in order to develop the fastest and most accurate MBO(N)D-II code both in terms of multibody dynamics (already done) and forcefield calculations, especially for large solvated systems (in plans). Note that our parent company SBI plans to invest in this Phase III effort.

Why do we need to impinge on the forcefield calculations? The current use of the CHARMm forcefield is not efficient for two reasons. First, CHARMm code [18] for calculation of electrostatic interactions is slow and memory-demanding due to historic reasons. Second, the way how MBO(N)D is interfaced to CHARMm now prevents us from unleashing the full potential of MTS. Note that the MTS can be implemented effectively only if there is a full control over the forcefield calculations (e.g. in order to use switching functions).

Due to all these memory and computational burdens, which currently exists for large solvated systems, there is a tendency to use relatively short cutoffs for electrostatics. For example, CHARMm only can be run with 10 Å cutoff (on SGI R10000 computers). Of course, any aggressive approximations greatly increase the speed but involve a loss of accuracy. To retain accuracy while computing electrostatic forces acting on each atom, one needs to sum up a very large amount of contributions from atoms within at least 15 Å sphere. MBO(N)D-II with OMBI and temporal-spatial MTS can provide substantial improvements over the current-state-of-the-art MD simulations, i.e. provide fast simulations of accurate (i.e. with very long cutoff) electrostatics.

During this Phase II work we have conceptualize a simple and, at the same time, technologically the most advanced approach to numerical treatment of multi-body molecular system, and this needs to be tuned to a water system. This approach involves a spatial-temporal decomposition of computations and then seamlessly "gluing" them through a symmetric scheme of weighting coefficients (what we showed to be a more general but simpler approach than RESPA [45]). It is clear that pair-wise interactions between atoms of water molecules can be integrated with different time scales and at each integration step computed with different level of detail (depending, mainly, on the distance between interacting waters).

We suggest a simple spatial decomposition, specific to the water system. First of all, the non-bond list should be based on water molecules rather than on atoms (H and O) like in CHARMm. Only this entails at least 9 times reduction in memory. This will also simplify the list building and operations with the list since no exclusion pairs (such as bonded atoms) exist. Electrostatic interactions between close waters (within 6 Å) should be performed atomistically. Electrostatic interactions between remote waters can be computed via dipoles (only one square-root operation is needed instead of nine!). Dipole forces are effectively converted into forces and torques for the rigid body water molecules. Switching functions are used (e.g. in 6-8 Å interval) to smoothly transform atomistic interactions between waters into dipole interactions, which is crucial for stable and accurate integration. It should be mentioned that for very remote interactions the waters can be grouped into clusters of two, three (or more) molecules so that the interactions can be further reduced to more "coarse" multipole calculations. The water "coarse-grain" model is naturally combined with rigid-body dynamics (OMBI) and with temporal MTS.

All the above spatial pair-wise computations (atomistic or "coarse") can be further decomposed in different time scales. We can develop better binning algorithm than the one which exists now in MBO(N)D. It will be based on a better utilization of chemical information, which was difficult to implement before since there was no control over the CHARMm forcefield. Solution can be based on evaluation of the second-order derivatives of the potential for each interaction and weighting it by mass.

From an implementation point of view, the advantage of our approach consists in the elimination of multiple non-bond lists for MTS bins. This includes "buffer" bins between the MTS bins and correspondingly the associated "buffer" lists as it is currently implemented in

CHARMm. All this possible due to a powerful scheme of symmetric weighting coefficients, which universally regulates different time scales, space groupings and smooth transitions between scales and space groupings (via switching functions convoluted with weighting coefficients).

The expected total speed-up is combined from a "spatial" factor and a "temporal" factor. We have estimated that the "spatial" factor can provide a speed-up, which varies from 5 to 20 (for 10-15 Å cutoffs) and up two-orders of magnitude for even longer cutoffs. The "temporal" factor can provide a speed-up, which varies from 2 to 5 (for 10-15 Å cutoffs) and up to one-order of magnitude for even longer cutoffs. The estimates are based on counting pair distributions as a function of distance and assuming that dipole model can be used starting from 6 Å (with 6-8 Å transition interval). It is also assumed that long-distance interactions (in 6-15 Å interval) can be integrated with 5-10 fs time steps. In overall, the temporal-spatial MTS can provide 10-100 speed-up over the state-of-the-art methods.

The implementation of the GUI for a stand-alone MD code (with the OMBI dynamics engine and the advanced forcefield computations) would be the next step in Phase III work.

The preparation of the code for commercial sale will be another major effort of Phase III. It should be noted that the current MBO(N)D code (MBO(N)D-I) is distributed by MSI. We might renegotiate the deal with MSI, since MBO(N)D-II is a quantum leap from MBO(N)D-I. Ultimately we could sell MBO(N)D-II as a stand-alone code with its own forcefield computations (as proposed above in this Phase III plan). It will be a business decision of our new parent company SBI on how to commercialize MBO(N)D-II. In the mean, SBI considers incorporation of MBO(N)D in its own production lines. In particular, large-scale MD simulations by MBO(N)D-II will be used in the generation of DynaPharm™ templates – virtual constructs of protein pharmacophores playing an important role in structure-based drug design.

5 Task 1: Complete the Implementation of High-Order Modified RKN Integrators into the Current Version of MBO(N)D

The key criterion in an efficient choice of time-stepping algorithm in MD is that it should allow the use of comparatively large time steps (or, ultimately, a favorable combination of multi time steps), while preserving accuracy and stability, and maintaining tractable levels of computational effort. In particular, a small number of forcefield (FF) evaluations is most desirable. Many numerical methods, such as explicit predictor-corrector methods and various implicit algorithms, have been used for solving the oscillatory stiff systems arising in MD simulations. Experience from discretized hyperbolic partial differential equations, which also often exhibit oscillatory stiffness, indicates that implicit methods effectively "damp" out the undesirable high-frequency dynamics and allow a longer time scale evolution to be computed. However, there is little analysis addressing the question of whether such large time step algorithms accurately simulate the real dynamics. Moreover in MD the evaluation of the potential is typically extremely expensive, so that many implicit schemes are prohibitively slow even with a longer time step. Perhaps as a consequence of this, the simple Verlet scheme currently remains as one of the most widely used algorithms. On the other hand, we believe that an investigation of mathematically more sophisticated integration methods within the context of practical MD simulations would be timely. The development of new conservative or symplectic time stepping schemes exploiting Hamiltonian systems of particular forms, such as separable, constrained, etc., is an extremely active area of research within the academic community. Moreover, surveys of current state of the art are now available (Refs. 3, 28, 36 and 51).

5.1 Lobatto Integrator as a Basis for High-Order Modified RKN Integrators

Moldyn's MBO(N)D (or MBO(N)D-I) code originated from NASA's general-purpose flexible multibody simulation program, DISCOS (Dynamics Interaction Simulation of Controls and Structures). It was developed into a more efficient Order (N) version (N-DISCOS) by Moldyn personnel. The original DISCOS code and its variants were capable of applying a number of integrators, including Runge-Kutta (Ref. 6), Gear (Ref. 63), and Adams (Ref. 62). However, what was suitable for multibody spacecraft and robotics problems was found to be inadequate for molecular modeling. In particular, Runge-Kutta requires four forcefield evaluations per step, compared to one per step for the Verlet type integrators. Furthermore, the high frequency content of the atomistic regions of a multigranular model require step-sizes on the order of 1-2 fs for stability.

Standard Verlet-type integrators, which are commonly used for molecular dynamics simulation, are not applicable to flexible multibody dynamics equations because those integrators operate under the assumption that acceleration terms are not dependent on velocity. In fact, the equations of motion contain terms that are quadratic in velocity, due to such contributions as gyroscopic terms, deformation-dependent inertia matrix, and kinematic constraints. A straightforward adaptation of Verlet-type integrators, by iterating on the velocity dependent terms (see Ref. 45 for iterated velocity Verlet, and Ref. 9 for Verlet central difference), was found not to be suitable for MBO(N)D's multibody equations. The linear and angular momenta exhibited a large amount of drift for those integrators.

The current integrator used within MBO(N)D is the Lobatto integrator with periodic stopping of the center of mass motion to counteract the drift in linear and angular momenta. With this velocity correction, the center of mass position of the ubiquitin example mentioned above is maintained to within 0.004 angstroms over a 10 ps simulation period.

The Lobatto integrator, very similar to the velocity Verlet integrator, consists of the following steps:

1. Propagate velocity to the half step and iterate till convergence:

$$v\left(\frac{\Delta t}{2}\right) = v(0) + a\left[q(0), v\left(\frac{\Delta t}{2}\right)\right]\frac{\Delta t}{2} \tag{5}$$

2. Propagate position to the full step:

$$q(\Delta t) = q(0) + v\left(\frac{\Delta t}{2}\right)\Delta t \tag{6}$$

3. Propagate velocity from the half step to the full step:

$$v(\Delta t) = v\left(\frac{\Delta t}{2}\right) + a\left[q(\Delta t), v\left(\frac{\Delta t}{2}\right)\right]\frac{\Delta t}{2} \tag{7}$$

The Lobatto integrator described above uses four acceleration calculations per step. The first of these involves initialization of the half-time velocity. The next two provide the iterative solution to Eq. (5), and the last acceleration calculation is used at the end of the step in Eq. (7).

The forcefield evaluations, on the other hand, are only needed at the beginning and end of each step, since these are only dependent on position, and not on velocity. Because the forces computed at the end of one step are used again at the beginning of the next step, the algorithm effectively needs to calculate forcefield evaluations only once per step. The CPU usage for calculating forcefield evaluations in the Lobatto integrator is therefore similar to that for Verlet-type integrators for molecular dynamics.

The symmetric architecture of the Lobatto integrator serves as a basis for deriving of higher-order symmetric RKN schemes described in the following Section.

5.2 New Parametrizations of Higher Order Time-Symmetric Integrators

In the Phase II project, we were able to finalize developments at Moldyn a new parametrization of a whole class of numerical integration algorithms based on modifying RKN integrators to accommodate velocity dependent terms. The new parametrization maintains the properties of time-reversibility and high-order (for increased accuracy), but requires more function evaluations per timestep. This is counterbalanced, however, by the ability to increase the timestep beyond the limits inherent in the second-order integrators. The advantages of these higher-order integrators will become apparent for highly substructured systems with low frequency dynamics content for which it is possible to take very large timesteps. We have shown that these integrators can offer better momentum conservation, for the same cost, than the second-order ones we have tested.

The Lobatto (modified RKN2) integrator utilized in MBO(N)D dynamics is accurate to order Δt^3, i.e., it is a second-order integrator. For many MD applications this accuracy is sufficient to provide a stable long-time simulation with an acceptable level of conservation in total energy and momenta. However, increasing the integrator's order is practically interesting from two points of view. First, in some simulations a higher accuracy may be desired, e.g., in simulating diffusion of water molecules into myoglobin where it is important to generate the exact constant energy path. In this case, the use of a more expensive (computationally) higher-order integrator may be justified. Second, even in cases with a relatively low specified level of accuracy, it is still interesting to investigate whether going to a higher-order integrator may provide better performance without requiring more CPU time. In other words, the question is whether the higher-order integrator with n forcefield evaluations per integration step of Δt performs better (or not) than the Lobatto integrator with one function evaluation per integration step of $\Delta t/n$. If the answer is positive then one could find the step for a higher-order integrator ($> \Delta t$) that will provide the same accuracy as the Lobatto integrator does but will require a smaller CPU usage. This means that the more complex structure of a higher-order integrator helps to "amplify" the usefulness of each forcefield evaluation since it involves an approximation over a few evaluation points. Intuitively, it is clear that this "amplification" effect will be more pronounced for molecular systems that exhibit dynamics with lower frequency content, a fact that also enables larger timesteps. This is observed in highly substructured molecules where large sets of atoms are grouped into rigid or flexible bodies (with low-frequency modes).

Given this practical interest in high-order integrators, high-order Lobatto-like integrators were produced following the framework of Suzuki (Ref. 54) which describes how to generate high-order time-reversible integrators from low-order ones. The results of Ref. 54 are formulated for systems with velocity-independent forces. Accordingly, the elementary low-order time-reversible integrator is the second-order Velocity Verlet (VV) integrator (although other forms of the second-order time-reversible integrators such as Verlet Position integrator may be assembled into a higher-order integrator). According to Ref. 54, the synthesized higher-order integrator is a time-reversible sequence of a few second-order integrators. Ref. 54 offers a general methodology to compute the parameters for the integrator's sequence (namely, the number of low-order integrators and the timesteps for each of them). This methodology was utilized to derive a high-order integrator for the system with velocity-dependent forces. In this case, the elementary second-order integrator is the Lobatto integrator (modified RKN2). The required assumption is straightforward and just states that the *velocity-dependent* Lobatto integrators may be combined in the same way as *velocity-independent* Verlet-type integrators in order to yield a higher-order integrator.

Significantly, new parameterizations were found for the integrator's sequence which provide a better time-sampling for the high-order integration of dynamical systems. In the Phase I work only the design of fourth-order time-reversible Lobatto-like integrators (also called fourth-order Lobatto or modified RKN4), for simplicity) was considered. However, the framework makes it possible to design higher-order integrators. It is also important to note that the high-order integrators developed so far are based on the Lobatto integrator, i.e., on a model-independent integrator. However, due to the generality of the presented framework, it is possible to use it for the design of a higher-order optimized multibody integrator. Using the second-order optimized multibody time-reversible integrator described in section 2.3 serves as the elementary integrator for this design.

The main idea behind the generation of the fourth-order integrator consists in the following (Ref. 54). The integrator is obtained by a symmetric repetition (product) of the second-order integrator in the form

$$U_{4\text{th}}(\Delta t) = U_{2\text{nd}}(\alpha\Delta t)U_{2\text{nd}}(\beta\Delta t)U_{2\text{nd}}(\alpha\Delta t) \tag{8}$$

where U is the integrator's operator, Δt is the integration step, and α and β are two real unknowns to be determined. Note that the integrator of Eq. (8) involves three right-hand side function evaluations per integration step.

The conditions for generating α and β which guarantee that the resulting integrator of Eq. (8) is of fourth-order may be obtained from the fundamental Baker-Campbell-Hausdorff (BCH) formula (Ref. 54). One of the corollaries of the BCH formula states that the multiplication of three exponential operators may be expressed in the form of a single exponential operator:

$$\mathbf{e}^{X}\,\mathbf{e}^{Y}\,\mathbf{e}^{X} = \mathbf{e}^{W} \tag{9}$$

where

$$\begin{aligned} W = {} & 2X + Y + \frac{1}{6}[Y,Y,X] - \frac{1}{6}[X,X,Y] + \\ & \frac{7}{360}[X,X,X,X,Y] - \frac{1}{360}[Y,Y,Y,X] + \ldots \end{aligned} \tag{10}$$

In Eq. (10) the commutator notation $[X,Y] = XY - YX$ is used, with higher-order commutators given as $[X,X,Y] = [X,[X,Y]]$.

The operator for the second-order Lobatto integrator results from the multiplication of three exponential operators (which is a result from the Trotter decomposition):

$$U_{2\text{nd}}(\Delta t) = \mathbf{e}^{A\frac{\Delta t}{2}}\,\mathbf{e}^{B\Delta t}\,\mathbf{e}^{A\frac{\Delta t}{2}} \tag{11}$$

where $A = a\left(x,v^{*}\right)\dfrac{\partial}{\partial v}$, $B = v\dfrac{\partial}{\partial x}$, a is the acceleration, x is the position, v is the velocity, and v^{*} is the midpoint velocity, i.e., the velocity at the middle of the integration step.

Applying the BCH-based formula of Eq. (9) to each second-order integrator in the product of Eq. (8), and accounting for the fact that each second-order integrator is a product of the velocity and position operators (see Eq. (11)), yields:

$$\begin{aligned} U_{2\text{nd}}(\alpha\Delta t) &= \mathbf{e}^{\alpha\Delta t\,\Lambda_1 + \alpha^3\Delta t^3\,\Lambda_3 + \alpha^5\Delta t^5\,\Lambda_5 + \ldots} \\ U_{2\text{nd}}(\beta\Delta t) &= \mathbf{e}^{\beta\Delta t\,\Lambda_1 + \beta^3\Delta t^3\,\Lambda_3 + \beta^5\Delta t^5\,\Lambda_5 + \ldots} \end{aligned} \tag{12}$$

where

$$\Lambda_1 = A + B,\quad \Lambda_3 = \frac{1}{12}[B,B,A] - \frac{1}{24}[A,A,B],\quad \Lambda_5 = \frac{7}{5760}[A,A,A,A,B] + \ldots \tag{13}$$

Applying Eq. (9) one more time, but now to the whole triple product of Eq. (8), and taking into account the expressions of Eq. (12), results in the following operator for the fourth-order integrator:

$$U_{4\text{th}}(\Delta t) = \mathbf{e}^{(\beta+2\alpha)\Delta t\,\Lambda_1 + \left(\beta^3+2\alpha^3\right)\Delta t^3\,\Lambda_3 + \left(\beta^5+2\alpha^5\right)\Delta t^5\,\Lambda_5 + \ldots} \tag{14}$$

In order for Eq. (11) to yield a fourth-order integrator (i.e., accurate up to Δt^5), two conditions must be satisfied:

$$\beta + 2\alpha = 0,\ \beta^3 + 2\alpha^3 = 0 \tag{15}$$

which ensure that $U_{4\text{th}}(\Delta t) = \mathbf{e}^{(A+B)\Delta t + O\left(\Delta t^5\right)}$. The unique real solution is obviously

$$\alpha = \frac{1}{2 - 2^{1/3}} \approx 1.3512, \quad \beta = \frac{2^{1/3}}{2 - 2^{1/3}} \approx 1.7024 \tag{16}$$

It is interesting to note that the general and simple framework of Ref. 54 leads to the same well-known fourth-order Runge-Kutta-Nystrom (RKN) integrator. We formulated the RKN4 (Lobatto) integrator to account for velocity-dependencies in the system forces. In Phase II we plan to implement promising high-order Lobatto integrators as well as optimized multibody integrators for MBO(N)D. Thereby, we will also generate higher-order integrators, e.g., the sixth-order integrator as a combination of the fourth-order integrators, etc. In the Phase I work we concentrated on the improvement of the parameterization for the fourth-order integrator.

The main disadvantage of the classical parameterization of Eq. (16) consists in the fact that the magnitude of both α and β are greater than 1, and therefore each of the two forward ($\alpha\Delta t$) and one backward ($\beta\Delta t$) substeps are substantially longer than the basis step (Δt). Figure 1 shows this by illustrating the triple product of Eq. (8) versus three second-order Lobatto steps over the time interval Δt. In other words, the second-order Lobatto (RKN2) and the fourth-order Lobatto (RKN4) both make three steps over the time interval Δt but RKN4 involves steps outside the integration interval, and therefore is more prone to instability at larger Δt values.

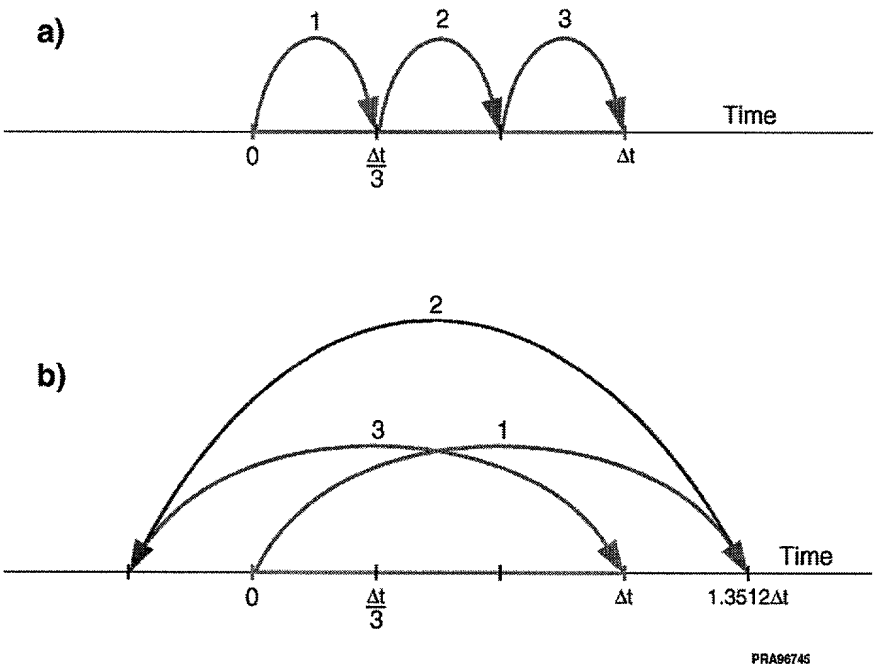


Figure 1. Three RKN2 Steps with Three Function Evaluations (a) Versus One RKN4 Step with Three Function Evaluations - Classical Parameterization (b).

We found a better parameterization for the fourth-order time-reversible integrator which involves only sampling that is internal to the time interval Δt. Since the goal is to make one RKN4 step with n function evaluations (in the above case n = 3) more efficient than n RKN2 steps (also using n function evaluations in all), one possible solution is to increase the number n.

Let us introduce the product of n ($n = 2m + 1$) second-order time-reversible integrators:

$$U_{4th}(\Delta t) = \left[\prod_{i=1}^{m} U_{2nd}^{(i)}(\alpha\Delta t)\right] U_{2nd}(\beta\Delta t) \left[\prod_{i=1}^{m} U_{2nd}^{(i)}(\alpha\Delta t)\right] \quad (17)$$

After multiple use of the BCH formula (similar to the general framework of Ref. 54 described above) it is possible to formulate the following conditions for the parameters α and β which guarantee that the operator $U_{4th}(\Delta t)$ is of fourth-order:

$$\beta+2m\alpha = 1, \quad \beta^3 + 2m\alpha^3 = 0 \quad (18)$$

Solution of Eq. (18) with $m = 2$ yields the following parameters:

$$\alpha = \approx 0.4144, \quad \beta = \approx -0.6579 \quad (19)$$

Figure 2 illustrates one RKN4 step with five function evaluations versus five RKN2 steps. One can see that all five sub-steps of the RKN4 are inside the time interval Δt. The most amazing fact is the following. Although the stepsize (Δt) for the new RKN4 with five functions evaluations must be larger than the stepsize (Δt) for the classical RKN4 with three function evaluations (with the ratio between the two Δt, is at least 5 : 3) in order to keep the same CPU cost of the new RKN4, the lengths of the forward and backward sub-steps have decreased. In other words, the new parameterization allows for a more uniform time-sampling without stepping outside the Δt interval. The new RKN4 integrator is therefore more stable than the classical RKN4 integrator. Given the fact that the classical parameterization has already been used for a few decades, our finding is quite significant.

The relative forward sub-step shown in the table above is defined as

$$l_F = \frac{\Delta t_F}{\Delta t_{RKN2}} = \frac{\alpha\,\Delta t}{\Delta t/n} = \alpha(2m+1) \quad (20)$$

and the relative backward sub-step is defined in a similar way:

The values of l_F and l_B are measures of internal stepsize. One can check by using different m integers that the classical time-reversible RKN4 (m = 1) has the largest (the most unfavorable) forward and backward sub-steps. The RKN4 with the new parameterization (m > 1) has smaller internal steps which entails a more uniform time-sampling. Thereby, the ratio l_F monotonously decreases with increasing m while the ratio l_B has a minimum at m = 4. The choice of m depends on the frequency content of a real system to be integrated. As was mentioned above, the higher-order integrators are most suitable for highly-substructured molecules with smooth low-frequency motion. Correspondingly, a larger value of m can be used for smoother motion. At this point in time, we have not implemented high-order time-reversible RKN integrators in MBO(N)D. Nevertheless, the implementation of the conventional fourth- and fifth-order Runge-Kutta integrators for highly substructured molecules indicate that it is worth going to higher-order integrators while discretizing smooth motion.

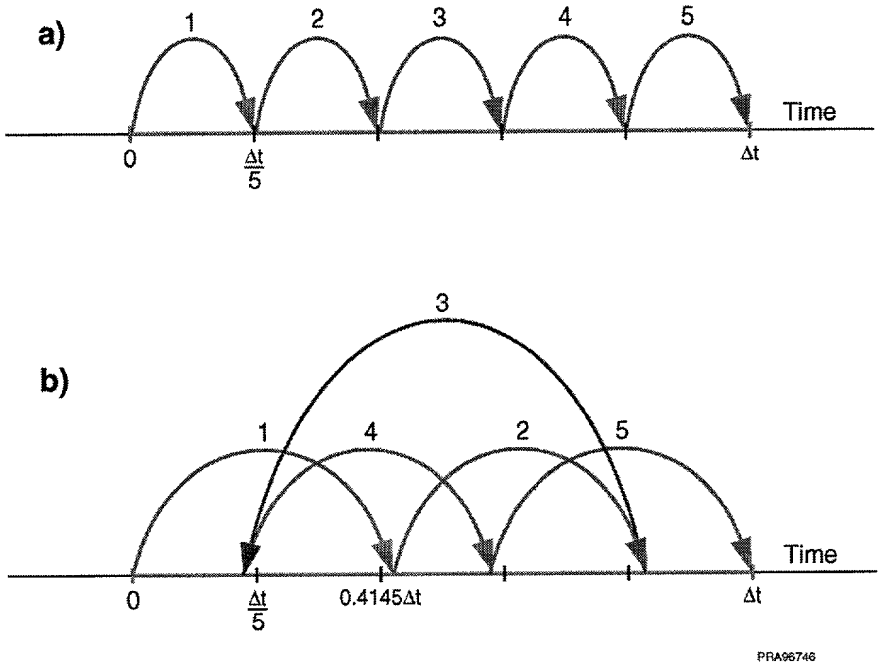


Figure 2. Five RKN2 Steps with Five Function Evaluations (a) Versus One RKN4 Step with Five Function Evaluations - New Parameterization (b).

$$I_B = \frac{\Delta t_B}{\Delta t_{RKN2}} = \frac{|\beta|\Delta t}{\Delta t/n} = |\beta|(2m+1) \tag{21}$$

5.3 Test Cases

During this Phase II reporting period, the fourth order Lobatto integrators corresponding to $m=1$ and $m=2$ were tested within the MBO(N)D program. The first integrator (with $m=1$, $\alpha \approx 1.3512$, $\beta \approx -1.7024$) is a sequence of three Lobatto steps and requires three energy function evaluations. For later discussion, we use the label "Lob4-3f" to describe this integrator, where the first number indicates the order of the integrator, and the second number indicates the number of forcefield evaluations. The second Lobatto variant (with $m=2$, $\alpha \approx 0.4144$, $\beta \approx -0.6579$) consists of five steps and five energy function evaluations. This variant is referred to as "Lob4-5f." Using the same naming convention, our basic Lobatto integrator is referred to as "Lob2-1f."

For a fair comparison of the three integrators, the time step for each integrator was selected so that the total number of forcefield evaluations is the same for each test case. Since the Lob2-1f integrator uses one evaluation per step, the Lob4-3f uses 3 calls per step, and the Lob4-5f uses 5 calls per step, the integration step sizes were chosen for the integrators in the ratio of 1:3:5.

Two molecular models, alanine dipeptide and a ribosomal protein fragment (1CTF in the Brookhaven Protein Databank), were selected for comparing the two versions of the fourth order Lobatto integrator with the existing Lobatto integrator. One substructuring scheme was used for alanine dipeptide, the molecule being relatively small. For 1CTF, three different substructuring schemes were tested. Conservation of energy and angular momentum were used as the criteria for judging the performance of the integrators.

Alanine Dipeptide The alanine dipeptide model was developed using the united atom model (i.e., with each non-polar hydrogen grouped with its neighboring heavy atom) and toph19 and parm19 parameters from CHARMm [18]. The initial coordinates were chosen to correspond to the deepest minimum of the energy potential. From these initial coordinates, the atomistic model was minimized, using 300 steps of Adapted Basis Newton-Raphson (ABNR). This was followed by equilibration using Langevin dynamics at 300K and friction parameter of FBETA=25 for 10 ps, with a 1 fs time step. The final coordinates and velocities were then used to start the MBO(N)D simulations for Runs 1 through 3. The MBO(N)D substructuring of alanine dipeptide treated each of the two peptide groups as rigid bodies, with each united atom methyl being modeled as individual particles (see Figure 3). MBO(N)D production simulations were carried out to 2 ns.

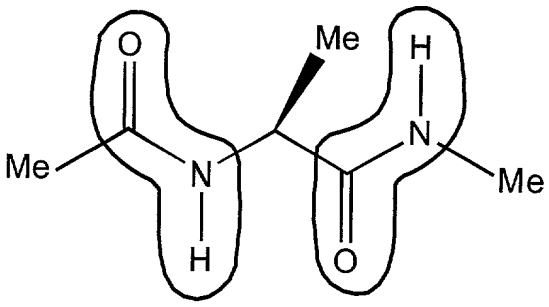


Figure 3. Substructuring used for alanine dipeptide model

We plotted total energy and angular momentum as a function of time, to evaluate the degree to which the different integrators are able to conserve those properties. Energy values are in units of kcal/mole. Angular momentum values are plotted in units of gram angstrom^2 per (mole-AKMA time unit). Linear momentum was not plotted, since the nonlinearity in the dynamics is mainly in the angular degrees of freedom, and not in the translational degrees of freedom. Moreover, previous studies had also shown that linear momentum is well conserved. Three runs of alanine dipeptide were made. In the first run, no center of mass stopping was performed. In the second and third runs, center of mass stopping was performed every 250 steps. In the third run, the time steps used were twice as long as that in the first two runs. Table 1 is a summary of the alanine dipeptide run conditions.

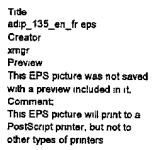
Table 1. Alanine Dipeptide Run Conditions and Observed Results

Run Number	Center of Mass Stopping	Time Steps[a] (fs)	Energy Conservation[b]			Momentum Conservation[b]		
			Best		WWorst	BBest		WWorst
1	No	1,3,5	21	45	43	45	43	21
2	Yes	1,3,5	21	45	43	21	45	43
3	Yes	2,6,10	21	45	5	21	45	1

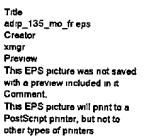
[a] Time steps are listed in the following order: Lob2-1f, Lob4-3f, Lob4-5f.

[b] The notations 21, 43, and 45 is used to represent Lob2-1f, Lob4-3f, and Lob4-5f, respectively

Alanine Dipeptide Run 1 The total energy plot in Figure 4(a) shows that Lob2-1f has the least amount of drift, although the fluctuation about its average value is highest among the three integrators tested. Lob4-5f has the least amount of total energy fluctuations, although it does exhibit a mild drift. Lob4-3f has the highest amount of total energy drift. The angular momentum plot of Figure 2b shows that the Lob4-5f integrator has the least amount of angular momentum error, with Lob4-3f being a close second. Lob2-1f has a larger amount of error, but the error seems to stabilize at a constant level after 1 ns.



(a)

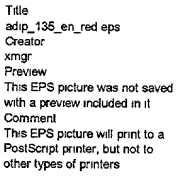


(b)

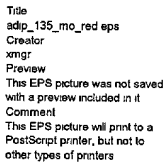
Figure 4. Alanine dipeptide Run 1: (a) total energy and (b) angular momentum. Center of mass motion is not stopped, resulting in energy and momentum drift.

Alanine Dipeptide Run 2 In this run, center of mass stopping was applied at a rate of once every 250 steps. That is, for the Lob2-1f integrator, it is applied every 250 fs, for the Lob4-3f integrator every 750 fs, and for the Lob4-5f integrator every 1.25 ps. The Lob2-1f integrator shows a reduction in the local fluctuations of the total energy, while the Lob4-5f integrator has a smaller reduction, and Lob4-3f shows little change (Figure 5(a)). Figure 5(b) shows a significant

improvement in the conservation of angular momentum for Lob2-1f, the peak errors having been reduced by a factor of 500 from that in Run 1. The improvement for the Lob4-3f and Lob4-5f integrators are factors of 30 and 40, respectively. The end result is that, for alanine dipeptide, the Lob2-1f integrator performs best in conservation of energy and momentum when center of mass stopping is applied.



(a)

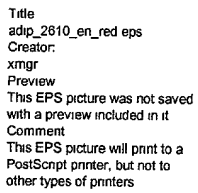


(b)

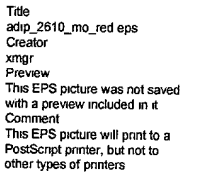
Figure 5. Alanine dipeptide Run 2: (a) total energy and (b) angular momentum. Center of mass motion is stopped, resulting in better momentum conservation. Energy drift is still evident in the Lob4-3f and Lob4-5f integrators.

Alanine Dipeptide Run 3 In Run 3, the integration step sizes are made larger by a factor of 2. Center of mass motion is still stopped at every 250th step. As shown in Figure 6a, there is a slight increase in the fluctuation of total energy for Lob2-1f, and the most significant degradation

is seen in Lob4-3f. The angular momentum error shows an increase over that in Run 2 by factors of 3-5, with the relative performance between the integrators being the same (Figure 6b).



(a)



(b)

Figure 6. Alanine dipeptide Run 3: (a) total energy and (b) angular momentum. Integration step-size is increased by a factor of 2, resulting in a significant degradation in Lob4-3f conservation properties.

Ribosomal Protein Fragment (1CTF) The initial structure was obtained from the Brookhaven Protein Databank. Polar hydrogen parameters (CHARMm version 19 parameters)

were employed resulting in a 596-atom model. Some charged side-chains of this protein model were then neutralized to compensate for the lack of solvent in the model. The Lennard-Jones and Coulomb interactions were treated by an atom-based cutoff of 9 Å with a smooth switching region of 2 Å. This structure was minimized for several hundred steps by the Adopted Basis Newton Raphson (ABNR) algorithm in CHARMm to remove artificial strains from the initial configuration. The atomistic model was heated from 0K to 290K over 5 ps, followed by equilibration for 40 ps at 290K. The coordinates and velocities of the molecule at the end of the atomistic equilibration were then used as the initial conditions for the MBO(N)D equilibration and production runs.

Three substructured models of 1CTF were used to test the integrators. These substructuring schemes were developed on a separate project, and are summarized in Table 2.

Table 2. The residues for each body definition within the secondary structure elements of 1CTF (note: the loops are considered part of the strands in this table).

	[1]Model A	[1,2]Model B	[2]Model C
Strand 1	2-5, 6-7, 8-10	←	←
Helix 1	11-13, 14-15, 16-17, 18, 19, **20**, 21-22, 23-25	11-12,**13**	14-17, 18-20
Helix 2	6-27, 28-29, 30-31, 32-33, 34, 35-37	**35**, 36-37	30-33, 34-37
Strand 2	38-42, 43, 44-45, 46-47	←	38-39, 40-42, 43-45
Helix 3	48-49, 50, 51-52, 53, **54**, 55-56, 57, **58**, 59-60, 61-62	←	50-54, 55-58
Strand 3	63-65, 66-68	←	63-64, 65-68

[1]Residues that had a side-chain body are shown in bold.

[2]Only modifications to Model A are shown for clarity: "←" means that the body definitions are same as those in Model A; only the *additional* body definitions are listed where appropriate (all other body definitions for that secondary structure are the same as in Model A). For example, in Model B/H1, the body definition 11-13 in Model A was split into the two bodies defined by residues 11-12 and 13, and all *other* body definitions in Model A/H1 are the same for Model B/H1. In Model C/H1, however, the bodies defined by 14-15, 16-17 in Model A/H1 have been joined into one body in Model C/H1 (14-17).

Model A is based on a side chain vector analysis of the atomistic simulation. This analysis suggested several residues that could be substructured such that the side chains of buried residues were mobile (shown in bold in Table 2). The body definitions suggested by this analysis were added to those in Model C. Model B is the same as Model A, except that more side chains bodies were added to test the effect of non-hydrophobic side-chain bodies, and additional side-chain bodies suggested by the side-chain analysis. Model C is based on a pseudo dihedral analysis, which suggested that the residues at the ends of the helices should be free, with additional bodies (based on intuition) added to the strands and helices to impart more motion.

For each substructured model, MBO(N)D equilibration was performed at 290K, with velocity re-scaling and center of mass stopping at every 100 steps. All equilibration runs were of 90 ps duration, except for Model C, where a 120 ps equilibration was performed. To perform the

tests discussed herein, MBO(N)D production runs (with constant energy conditions) were initialized from the final coordinates and velocities of the equilibration runs.

Runs were made wherein the run conditions of center of mass stopping and integration step size were varied, similar to the alanine dipeptide test cases. In addition, we tested the integrators on several different substructuring schemes. We plotted the percent error in total energy relative to its initial value, and the angular momentum.

Table 3 summarizes the run conditions and observed results of the 1CTF test cases.

Table 3. 1CTF Run Conditions and Observed Results

Run Number	Model	Center of Mass Stopping	Time Steps[a] (fs)	Energy Conservation[b]			Momentum Conservation[b]		
				eBest		Worst	Best		Worst
1	A	No	2,*,10	45		21	45		21
2	A	Yes	2,6,10	45	21	43	45	21	43
3	A	Yes	4,12,20	21	45	43	21	45	43
4	B	Yes	2,*,10	45		21	45		21
5[c]	C	Yes	2,*,10	45		21	21	2	45

[a]Time steps are listed in the following order: Lob2-1f, Lob4-3f, Lob4-5f; an asterisk indicates the corresponding integrator was not used in the run.

[b]The notations 21, 43, and 45 is used to represent Lob2-1f, Lob4-3f, and Lob4-5f, respectively; the Lob4-3f integrator was used for only some of the runs.

[c]The differences in performance between the integrators were very slight for this run.

The general conclusion from the test cases is that, for small integration steps, the Lob4-5f integrator (which we developed) performs better than the Lob2-1f integrator, while the classical Lob4-3f integrator (RKN4 in the open literature) performs worse than the Lob2-1f integrator. The poor behavior of the Lob4-3f integrator can be traced to its sampling scheme, which involves derivative calculations outside the current time step interval. This comparative performance is in agreement with the results of the Matlab pilot plant tests of Phase I. It also appears that, for larger step sizes, the Lob2-1f integrator is the best model-independent integrator for the multibody dynamics representation of molecular systems that is embodied in MBO(N)D.

6 Task 2: Complete the Formulation of OMBI for the Type of Chains of Rigid/Flexible Bodies Found in SMD

The integrators developed and modified in Task 1 (see Section 5), including the newly developed high-order RKN Lobatto-type integrators, are model-independent, i.e., they can be directly implemented for an arbitrary set of dynamical equations. The only requirement on the structure of the equations in the case of the RKN integrators is the fact that the system should be of second-order (i.e., with an explicit partition of the state vector into position and velocity states.) Thereby, the RKN integrators account for the velocity-dependency, namely for the dependency of accelerations not only on the position states but also on the velocity states. Note that the RK (unlike RKN) integrator does not use any of the specializations.

Such generality of the integrators made the implementation to MBO(N)D dynamics relatively straightforward. This was beneficial in the sense that such complicated issues as incorporation of constraints in multi-body dynamics did not need to be considered for the integrator design due to the fact that the MBO(N)D formulation in the relative (hinge) coordinates automatically ensures the maintenance of the constraints with machine accuracy (the EXACT constraints formulation (Ref. 60,61) for tree topologies.

After implementing the time-reversible modified RKN integrators in MBO(N)D we concluded that these integrators provide an excellent performance (in terms of conserving the invariants of motion such as total energy and momenta) for long-time MD simulations. All non-reversible integrators (as known in the literature and observed by us in MBO(N)D) exhibit a substantial systematic drift in the total energy and momenta. Note that this drift may be made smaller by using a smaller timestep. But, first, this will significantly slow down the MD simulations, and, second, the drift is still systematic (e.g., linear as for RK4) and this will sooner or later degrade performance in long-time simulations. The most likely avenues for optimizing the integrator structure and coefficients have already been taken for the integration of arbitrary second-order systems of equations with velocity-dependent terms. Indeed, as was mentioned in Section 5.2, the Lobatto-type parameterization based on using the mid-step velocity provides the best (up to second-order) cancellation of the errors in the Taylor expansion of the system's accelerations. In some cases, the higher-order RKN may perform better than Lobatto but from the theoretical point of view they only entail an optimal sequence of the RKN2 (Lobatto). It should also be mentioned that using implicit integration schemes (both on velocity and position states) is not suitable for MD due to the high computational cost of force-field evaluations.

In this situation, any further improvement in the integration of MBO(N)D dynamics appears to be in combining integrator design and formulation of equations of motion. In other words, the objective is to find such a formulation that the integrator designed for this particular formulation will be efficient both in terms of accuracy and speed. Multi-body dynamics are characterized by a quadratic dependency of the accelerations on velocities, which is due to such contributions as gyroscopic forces, deformation-dependent inertia matrix, and kinematics. The question is whether it is possible to treat analytically the quadratic velocity factor in an explicit integration scheme.

We decided to approach this problem stage-by-stage, since the benefits of the model-independent integrators (such as applicability to arbitrary dynamical equations) are not valid any more. For example, the issue of structural flexibility or constraints becomes directly involved in the design of the optimized multibody integrator. Correspondingly, the first stage deals with the case of unconstrained rigid body dynamics in absolute coordinates. The second stage provides generalizations on how to deal with structural flexibility and the third stage accounts for constraints.

6.1 Derivation of the Optimized Multibody Integrator (OMBI) for SMD

The derivation of the OMBI is based on the Liouville-Trotter formalism (Ref. 45), which enables time-symmetric decomposition of equations of motion with respect to the position and velocity states as well as with respect to different DOF of the system. In the theory of differential equations this formalism is also evidenced as a method of symmetric successive freezing of the system's variables. Each of the systems arising as a result of this decomposition is the subject for analytical integration within the time step at which decomposition was performed. In this way, the Liouville-Trotter mechanism provides the framework for integrating complex nonlinear systems for which obtaining analytical solutions would be impossible. The resulting integrator is still numerical but with the built-in analytical solutions for the decomposed systems. Due to the time-symmetric nature of the Trotter decomposition, the integrator is time-reversible which results in high performance during long-time simulations. Since the Trotter

decomposition still involves some approximation, the art of designing the optimized multibody integrator consists in finding such a formulation for the equations of motion that has the largest degree of natural decomposition in the system's variables and is rich in its analytical properties. It was found that the formulation using Euler parameters and momentum variables, with the definition of the momentum in the body frame, is the most suitable formulation, which meets the above requirements. We generalized the Liouville-Trotter formalism for this formulation to deal with the fact that the equations of motion are not of second-order.

In implementing the Liouville-Trotter formalism we exploited the analytical advantages of the formulation with Euler parameters and momentum variables. First of all, the definition of the momentum in the body-fixed axis system uncouples position and velocity states in the equation for momentum as much as possible. This makes the Trotter decomposition more accurate. In addition, the fact that all functional dependencies in the equations of motion for this formulation are either linear (e.g., kinematics for Euler parameters) or quadratic (e.g., gyroscopic coupling effects) was exploited. Correspondingly, in the first case the matrix exponential is needed to propagate the states of the linear system. In the second case, we implemented an additional Trotter decomposition of a vector differential equation and solved the resulting scalar Riccati-type equations analytically at the integration time step.

The detailed form of the OMBI for a general case, which includes rotational, translational and deformational (due to flexibility) DOF is given in the subsections below.

6.1.1 Equations of Multibody Dynamics

The equations of motion with the Euler parameters parametrization for rotations are represented in the form

$$\begin{aligned} \dot{q} &= V(q, p) \\ \dot{p} &= G(q, p) \end{aligned} \tag{22}$$

Here, the $\left(7N + \sum_{i=1}^{N} m_i\right) \times 1$ state vector q consists of N blocks, where N is the number of bodies in the system and m_i is the number of modes in the i-th body:

$$q = \left(\begin{array}{c} q_1 \\ \hline q_2 \\ \hline \vdots \\ \hline q_N \end{array}\right) \tag{23}$$

Each block of the vector q has the following structure (the index i which numbers the bodies in the system is omitted for simplicity):

$$q_i = \left(\begin{array}{c} e \\ \hline x \\ \hline \xi \end{array}\right) \tag{24}$$

where e is the 4×1 vector of Euler parameters, x is the 3×1 vector which represents the translational motion of the body's center of mass (COM), and ξ is the $m_i \times 1$ vector of modal coordinates which model the body's deformation.

The vector p is the $\left(6N+\sum_{i=1}^{N} m_i\right)\times 1$ vector of momenta which also comprises N blocks (for each body in the system):

$$p=\begin{pmatrix} p_1 \\ \hline p_2 \\ \hline \vdots \\ \hline p_N \end{pmatrix} \tag{25}$$

where each block is defined by an equation:

$$\begin{pmatrix} p_\omega \\ \hline p_u \\ \hline p_\xi \end{pmatrix} = M(\xi)\begin{pmatrix} \omega \\ \hline u \\ \hline U_\xi \end{pmatrix} \tag{26}$$

Here, the mass matrix of the i-th body is expressed in a general non-diagonal block form as

$$M(\xi) = \left[\begin{array}{c|c|c} I(\xi) & S(\xi) & d(\xi) \\ \hline S(\xi) & M & a(\xi) \\ \hline d^T(\xi) & a^T(\xi) & e(\xi) \end{array}\right] \tag{27}$$

where $I(\xi)$ is the 3 x 3 inertial tensor of the body, M is the 3x3 diagonal mass matrix of the body, $e(\xi)$ is the $m_i x m_i$ modal mass matrix , and the blocks S, d, and a represent the corresponding couplings between the diagonal blocks of the mass matrix $M(\xi)$. A general dependency of the body's matrix on its deformation is formalized by a dependency on the modal coordinates ξ. (See Refs. 66, 67 for details).

The vector of absolute velocities for the i-th body has the following block structure:

$$U_i = \begin{pmatrix} \omega \\ \hline u \\ \hline U_\xi \end{pmatrix} = \begin{pmatrix} \omega_x \\ \omega_y \\ \hline \omega_z \\ \hline u \\ v \\ \hline w \\ \hline U_{\xi_1} \\ \vdots \\ U_{\xi_m} \end{pmatrix} \tag{28}$$

where ω is the 3 x 1 vector of the body's angular velocities projected onto the body frame, u is the 3 x 1 vector of the translational velocities of the body's COM projected onto the body frame, and U_ξ is the $m_i \times 1$ vector of modal velocities.

The $\left(6N+\sum_{i=1}^{N} m_i\right)\times 1$ vector $G(q,p)$ formalizes all forces acting on each generalized coordinate. This vector is defined by

$$G = G_{FF} + \tilde{\Omega} M U + \frac{1}{2} U^T \left[M_{,j} \right] U \tag{29}$$

where the first term, $G_{FF}(q)$, represents the contribution from forcefield interactions. The second term accounts for gyroscopic coupling effects (the matrix $\tilde{\Omega}$ is an assembly of three skew matrices for the rotational and translational velocity vectors which models the coupling of rotational and translational motion). The third term accounts for forces due to the deformation-dependency of the mass matrix (thereby, what is meant by $\left[M_{,j} \right]$ is the partial derivative of each element of $M(\xi)$ with respect to the *j*-th generalized coordinate). The functional dependency $V(q, p)$ represents the kinematical ties between the system's generalized coordinates and velocities corresponding to the Euler parameter formulation of multibody dynamics.

Taking the above into account we will use the following block-vector form of the equations of motion for each body in the system (omitting for simplicity the index *I* which numbers bodies):

$$\begin{aligned} \dot{e} &= W(\omega)\, \mathrm{e} \\ \dot{x} &= C(e) u \\ \dot{\xi} &= U_{\xi} \\ \dot{p}_{\omega} &= G_{\omega}(q) + p^T \left[A_{\omega} \right] p \\ \dot{p}_{u} &= G_{u}(q) + p^T \left[A_{u} \right] p \\ \dot{p}_{\xi} &= G_{\xi}(q) + p^T \left[A_{\xi} \right] p \end{aligned} \tag{30}$$

Here, the matrices $W(\omega)$ and $C(e)$ represent the kinematical relations and are written in the form:

$$W(\omega) = \frac{1}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & -\omega_y & -\omega_x & 0 \end{bmatrix} \tag{31}$$

$$C(e) = \begin{bmatrix} 1 - 2\left(e_2^2 + e_3^2\right) & 2(e_1 e_2 + e_3 e_0) & 2(e_1 e_3 - e_2 e_0) \\ 2(e_2 e_1 - e_3 e_0) & 1 - 2\left(e_3^2 + e_1^2\right) & 2(e_2 e_3 + e_1 e_0) \\ 2(e_3 e_1 + e_2 e_0) & 2(e_3 e_2 - e_1 e_0) & 1 - 2\left(e_1^2 + e_2^2\right) \end{bmatrix} \tag{32}$$

Note that the matrix $W(\omega)$ has been evaluated by a linear transformation of the original Euler parameter kinematics equation $\dot{e} = Q(e)\omega$ so that $W(\omega) e = Q(e)\omega$. The matrix $C(e)$ is a rotation matrix from the body frame to the inertial frame. The tensors $\left[A_{\omega} \right]$, $\left[A_{u} \right]$, and $\left[A_{\xi} \right]$ symbolically represent quadratic dependency on the velocity (momentum) factor in the equations of multibody dynamics. The computational scheme for realization of $p^T \left[A \right] p$ accounts for sparsity of the tensor $\left[A \right]$. (See Refs. 66, 67 for details).

It should also be mentioned that in the above dynamical equations the transformation of the momentum vector p into the velocity vector U is realized by a solving a system of linear equations $p = M(\xi)U$.

6.1.2 Liouville-Trotter Formalism

The derivation of the OMBI is based on the Lioville-Trotter formalism (RESPA-type approach, see Ref. 45). The Lioville-Trotter formalism decomposes the original system of differential equations into a number of smaller subsystems integration of which is simpler than integration of the original system (for example, each subsystem can be integrated analytically). The important property of the Trotter decomposition consists in the fact that it is time-symmetric which results in high accuracy of the resulting integration algorithm.

For the case of dynamics of a system of flexible bodies the following Trotter decomposition may be used:

$$U(t) = \mathbf{e}^{iL\Delta t} = \mathbf{e}^{iL_p\frac{\Delta t}{2}}\ \mathbf{e}^{iL_q\Delta t}\mathbf{e}^{iL_p\frac{\Delta t}{2}} + O(\Delta t^3) \tag{33}$$

where the position Liouville operator is defined as

$$iL_q = \dot{q}\frac{\partial}{\partial q} \tag{34}$$

and the momentum Liouville operator as

$$iL_p = \dot{p}\frac{\partial}{\partial p} \tag{35}$$

In its turn the position Liouville operator may be decomposed into parts corresponding to the rigid (z) and flexible (ξ) degrees of freedom (DOF):

$$iL_q = iL_z + iL_\xi = \dot{z}\frac{\partial}{\partial z} + \dot{\xi}\frac{\partial}{\partial \xi} \tag{36}$$

Correspondingly, the following Trotter decomposition may be performed as:

$$\mathbf{e}^{iL_q\Delta t} = \mathbf{e}^{iL_z\frac{\Delta t}{2}}\mathbf{e}^{iL_\xi\Delta t}\mathbf{e}^{iL_z\frac{\Delta t}{2}} + O\left(\Delta t^3\right) \tag{37}$$

It is important to note that other Trotter decompositions are possible in order to obtain a time-symmetric integrator. For example, the system may be decomposed first with respect to rigid and flexible DOF and only then each sub-system can be decomposed with respect to position and momentum states. This decomposition makes it possible to utilize the analytical solution for harmonic oscillator (in the case when flexibility is modeled by using a linearizied system of equations with the corresponding stiffness matrix). However, the drawback of this Trotter decomposition is in the fact that two evaluations of the forcefield are needed per integration step (each forcefield evaluation corresponds to the previous and updated deformations of the body). The present Trotter decompositions of Eq. (30) are advantageous in terms that only one position-dependent operation per step is needed. This includes one forcefield evaluation and one factorization of the mass matrix $M(\xi)$ to solve $M(\xi)U = p$ for U. Moreover, the chosen decomposition is suitable for treating the body's flexibility in a general (not necessarily linear) form.

6.1.3 Description of the Five Stage OMBI

According to the Trotter decomposition presented above the Optimized Multibody Integrator (OMBI) includes five stages, which formalize the following operations.

Stage 1. Propagate the momentum vector to the half step $t = \Delta t/2$ freezing the position states at $t = 0$:

$$\dot{p} = G(q_0, p), \quad t \in \left(0, \frac{\Delta t}{2}\right) \tag{38}$$

with initial conditions $p(0) = p_0$.

Stage 2. Propagate the vector of rigid position variables to the half step $t = \Delta t/2$ freezing the flexible position variables at $t = 0$ and the momentum vector at $t = \Delta t/2$:

$$\dot{z} = V_z(z, \xi_0, p_{1/2}), \quad t \in \left(0, \frac{\Delta t}{2}\right) \tag{39}$$

with initial conditions $z(0) = z_0$.

Stage 3. Propagate the vector of flexible position variables to the full step $t = \Delta t$ using momenta (recomputed in modal velocities) at $t = \Delta t/2$:

$$\dot{\xi} = U_{\xi_{1/2}}, \quad t \in (0, \Delta t) \tag{40}$$

with initial conditions $\xi(0) = \xi_0$.

Stage 4. Propagate the vector of rigid position variables to the full step $t = \Delta t$ freezing the flexible position variables at $t = \Delta t$ and the momentum vector at $t = \Delta t/2$:

$$\dot{z} = V_z(z, \xi_1, p_{1/2}), \quad t \in \left(\frac{\Delta t}{2}, \Delta t\right) \tag{41}$$

with initial conditions $z\left(\frac{\Delta t}{2}\right) = z_{1/2}$.

Stage 5. Propagate the momentum vector to the full step $t = \Delta t$ freezing the position states at $t = \Delta t$:

$$\dot{p} = G(q_1, p), \quad t \in \left(\frac{\Delta t}{2}, \Delta t\right) \tag{42}$$

with initial conditions $p\left(\frac{\Delta t}{2}\right) = p_{1/2}$.

It is important to note that for the particular case of ideal rigid bodies ($\xi \equiv 0$) the OMBI is comprised of only 3 stages since stages 2, 3, and 4 become unified in one single stage (to propagate the vector of rigid positions to the full step). In the general flexible case the body is symmetrically rigidized over each half of the time step. In this way, as was mentioned above, the particular Trotter decomposition used for the derivation of the OMBI ensures that all position-dependent operations, such as forcefield evaluations are performed only once at each integration step (assuming that the position-dependent operation at the beginning of the step coincides with the operation at the end of the previous step). From a physical point of view, this integration scheme results in position-dependent operations being performed for the bodies,

which are in their initial deformation at the beginning of the step and in their final deformation at the end of the step.

The following subsection provides concretizations of the OMBI stages by describing how the integration of each subsystem is carried out, either analytically or numerically. Note that the following derivations are the heart of the OMBI since they define the efficiency of the integration process and require some effort in finding elegant analytical solutions.

6.1.4 Concretization of Stages 2 and 4 (Kinematics of the Rigidized Body)

These stages are defined by similar equations (38) and (39). All notations here correspond to Eq. (39) i.e. for Stage 2. Note that for Stage 4 one has to make the substitutions $\xi_0 \to \xi_1$ and $\left(0, \frac{\Delta t}{2}\right) \to \left(\frac{\Delta t}{2}, \Delta t\right)$. The equations in Stages 2 and 4 define the kinematics of the rigidized body with velocities frozen at the midpoint of the time interval and have the form (in block-vector notations):

$$\begin{aligned} \dot{e} &= W(\omega_{1/2})\, e \\ \dot{x} &= C(e)\, u_{1/2} \end{aligned} \tag{43}$$

First, the equation for propagating the Euler parameters, to the half step is considered. This is a linear equation and its solution is based on the matrix exponential:

$$e_{1/2} = E(\omega_{1/2})\, e_0 \tag{44}$$

An elegant form of this matrix exponential E was found. The derivation of the matrix exponential is based on its Taylor expansion with further analytical summation of the series exploiting the fact that the matrix W is skew-symmetric. To the best of our knowledge, this result appears to be new in the literature. According to our derivations, the matrix exponent can be written in the form:

$$E(\omega_{1/2}) = \mathbf{e}^{W(\omega_{1/2})\frac{\Delta t}{2}} = \cos(\gamma \Delta t) I + \sin(\gamma \Delta t) D \tag{45}$$

where

$$\gamma = \frac{1}{4} |\omega_{1/2}| \tag{46}$$

$$|\omega_{1/2}| = \left[\omega_x^2 + \omega_y^2 + \omega_z^2\right]^{1/2}_{t=\frac{\Delta t}{2}} \tag{47}$$

is the magnitude of the vector of angular velocities ω (at the half step) projected onto the body frame,

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{48}$$

is the identity matrix,

$$D = \begin{bmatrix} 0 & \bar{\omega}_x & \bar{\omega}_y & \bar{\omega}_z \\ \bar{\omega}_x & 0 & \bar{\omega}_z & \bar{\omega}_y \\ \bar{\omega}_y & \bar{\omega}_z & 0 & \bar{\omega}_x \\ \bar{\omega}_z & \bar{\omega}_y & \bar{\omega}_x & 0 \end{bmatrix} \tag{49}$$

is a skew-symmetric matrix of the vector of directional cosines. Note that the directional cosines $\bar{\omega}$ define the orientation of the vector of angular velocities ω in the body frame and computed as

$$\bar{\omega} = \frac{\omega_{1/2}}{|\omega_{1/2}|} \tag{50}$$

The second step in realizing the kinematics equations is to propagate the translational coordinates for the body's COM. The corresponding propagator was expressed in a very simple analytical form (using the analytical solution for the matrix exponent of Eq. (45) and the quadratic dependency of the rotation matrix $C(e)$ on the vector of Euler parameters e, see Eq. (32)):

$$x_{1/2} = x_0 + \Lambda u_{1/2} \tag{51}$$

where

$$\Lambda = \begin{bmatrix} 1-2(\Delta_{22}+\Delta_{33}) & 2(\Delta_{12}-\Delta_{03}) & 2(\Delta_{13}+\Delta_{02}) \\ 2(\Delta_{12}+\Delta_{03}) & 1-2(\Delta_{11}+\Delta_{13}) & 2(\Delta_{23}-\Delta_{01}) \\ 2(\Delta_{13}-\Delta_{02}) & 2(\Delta_{23}+\Delta_{01}) & 1-2(\Delta_{11}+\Delta_{22}) \end{bmatrix} \tag{52}$$

$$\Delta = \mu_0 \left[e_0 e_0^T \right] + \mu_{CS} \left[e_0 b_0^T + b_0 e_0^T \right] + \mu_S \left[b_0 b_0^T \right] \tag{53}$$

$$b_0 = D\, e_0 \tag{54}$$

$$\mu_c = \frac{\Delta t}{4} + \frac{\sin(2\gamma\Delta t)}{8\gamma} \tag{55}$$

$$\mu_s = \frac{\Delta t}{4} - \frac{\sin(2\gamma\Delta t)}{8\gamma} \tag{56}$$

$$\mu_{cs} = \frac{\left[\sin(\gamma\Delta t)\right]^2}{4\gamma} \tag{57}$$

6.1.5 Important Property of the OMBI's Propagator for Euler Parameters

The OMBI's propagator of Eq. (44) for Euler parameters has an important practical property which consists in the fact that the propagator ensures exact (up to machine accuracy) maintenance of the constraint $|e| = 1$.

To prove this fact, it is sufficient to show that the matrix exponential $\mathbf{e}^W$ (where $W = W(\omega_{1/2})\frac{\Delta t}{2}$ for simplicity) is an unitary matrix and, thus, retains the length of the vector e: $|e_{1/2}| = |e_0|$. The proof is based on the following two equalities:

$$\left[\mathbf{e}^W\right]^{-1} = \mathbf{e}^W \tag{58}$$

which is due to the property of stationarity of the differential equation for e (assuming that $W(\omega_{1/2})$ is "frozen"), and

$$-W = W^T \tag{59}$$

which is due to the fact that W is skew-symmetric. Based on the above two equalities it is easy to show that

$$\left[\mathbf{e}^{W}\right]^{-1} = \mathbf{e}^{-W} = \mathbf{e}^{W^T} = \left[\mathbf{e}^{W}\right]^{T} \Rightarrow \left|e_{1/2}\right| = \left|e_0\right| \tag{60}$$

Our tests during the Phase II work confirmed that the OMBI maintains the constraint on the length of the Euler parameter vector up to machine accuracy over hundreds of thousands of time steps and beyond (even using large time steps within the stability limits).

It should be noted that conventional (non-model based) integrators (e.g., Runge-Kutta etc.) involve a certain truncation error which quickly results in a build-up of error in the vector of Euler parameters e (at hundreds of time steps, even if those steps are relatively small). Usually, some artificial normalization of the Euler parameter vector are used to maintain the constraint $|e|=1$. The OMBI, which is directly tailored for the equations of multibody dynamics, easily obviates this difficulty and makes Euler parameters even more attractive for these applications.

6.1.6 Concretization of Stage 3 (Modal Kinematics)

The realization of stage 3, i.e. the propagation of the vector of modal coordinates to the full step, is trivial (due to frozen $U_{\xi_{1/2}}$ in Eq. (40)):

$$\xi_1 = \xi_0 + U_{\xi_{1/2}} \Delta t \tag{61}$$

6.1.7 Concretization of Stages 1 and 5 (Momentum Equations)

The realization of stages 1 and 5 involves solving the Riccati-type (i.e., quadratic) equation for the momentum vector p:

$$\dot{p} = G(q_0) + p^T [A] p \tag{62}$$

First of all, it is important to stress that this equation is for a general case of flexible bodies and its analytical solution over an arbitrary time interval is problematic. Note that the analytical solution remains difficult even for the case of rigid-body torque-free rotation when the quadratic dependency on momentum takes a particular form $p \times (I^{-1} p)$ and $G = 0$. In the latter case the solution may be expressed in terms of the Jacobi elliptic functions (Ref. 32) (which require tabulation). That is why it was decided not to search for a general analytical solution of Eq. (62) in closed vector form for an arbitrary time interval, but instead to find a numerical algorithm for solving Eq. (62) at the time step Δt.

We developed a scalar version of the propagator for Eq. (62). This version is based on further time-symmetric decomposition of the Liouville operator with respect to each DOF. A general idea of this decomposition may be illustrated by the following example. Let the integration of the n-dimensional system of Eq. (62) be performed at the half step in accordance with the following Liouville-Trotter formalism:

$$e^{\left(\sum_{i=1}^{n} iL_i\right)\frac{\Delta t}{2}} = \left[\prod_{i=1}^{n-1} e^{iL_i\frac{\Delta t}{4}}\right]\left[e^{iL_n\frac{\Delta t}{2}}\right]\left[\prod_{i=1}^{n-1} e^{iL_i\frac{\Delta t}{4}}\right] + o(\Delta t^3) \tag{63}$$

From a physical point of view, Eq. (63) entails that the 1st DOF is integrated over a quarter of the step with other DOF being frozen, then the 2nd DOF is integrated at the quarter of the step while the other DOF are frozen and so on. This process is continued until the (n-1)th DOF is integrated. Then the n-th DOF is integrated at the half step (all other DOF are being kept frozen). After that the process of integrating the first (n-1) DOF over a quarter of the time step is repeated in reverse order (starting from the (n-1)th DOF). As can be seen, the above computational cycle is time-symmetric.

The realization of the operation for each DOF can be considered as the integration of a scalar Riccati equation

$$\dot{p} = ap^2 + bp + c \tag{64}$$

Eq. (64) can be easily solved through transformation into a two-dimensional Hamiltonian system.

The final result can be written in the closed form:

$$p(t) = \begin{cases} \dfrac{\sqrt{\Delta}\tan\left[\tan^{-1}\dfrac{2ap(0)+b}{\sqrt{\Delta}} + \dfrac{\sqrt{\Delta}}{2}t\right] - b}{2a} & \text{if } \Delta > 0 \\ \dfrac{\sqrt{\Delta}\tanh\left[\tanh^{-1}\dfrac{2ap(0)+b}{\sqrt{-\Delta}} - \dfrac{\sqrt{-\Delta}}{2}t\right] - b}{2a} & \text{if } \Delta < 0 \end{cases} \tag{65}$$

where $\Delta = 4ac - b^2$

This algorithm can be easily programmed, by forming coefficients a, b, and c for each scalar differential equations and then using Eq. (65) to propagate the current DOF forward in time. The simulations in MBO(N)D demonstrate that the propagator of Eq. (65) exactly presumes the time symmetry of the entire integration process.

6.2 Implementation of OMBI for Unconstrained Particles and Rigid Bodies

The main objective here is to implement OMBI (Optimal Multi-Body Integrator), i.e. formulate it in a form of requirements for the design of the OMBI code in the OO C++ MBO(N)D (MBO(N)D-II). OMBI is formulated in Section 6.1 for the case of multi-body dynamics in the absolute coordinate system. The OMBI integrator be first implemented for the case when: 1) the multi-body system includes particles and rigid bodies; 2) rotation of each rigid body is modeled by four Euler parameters; 3) momenta (not velocities) are used as the state-vector elements; and 4) there are no inter-body constraints (only inter-body interactions).

The OMBI algorithm propagates dynamics of interacting particles and rigid bodies at a single integration step, Δt. OMBI, designed to provide optimal accuracy and stability of integration, is tailored for a particular formulation of unconstrained multibody dynamics, which uses Euler parameters and momenta (for rigid bodies). Correspondingly, the integration process is performed in an explicit form.

The OMBI algorithm (for particles and rigid bodies) described in this document contains reusable functions, which are also utilized in the OMBI algorithm (with flexible bodies) and in the

OMBI algorithm with SHAKE-type constraints. These functions are described in a form amenable for their reuse.

6.2.1 Dynamics of a Particle

Unlike the Lobatto integrator, which is a general purpose algorithm for a second-order system, OMBI is tailored to a particular form of dynamic equations. This is especially important for integrating dynamics of rigid bodies. However, OMBI for particles should follow this formalism for the sake of universality. Correspondingly, the dynamics of a particle are described in the following form:

$$\begin{aligned} \dot{q} &= V(p) \\ \dot{p} &= \left[G_{FF}^{abs} \ (q) \right]_f \end{aligned} \tag{66}$$

Note that Eq. (66) uses momenta rather than velocities (as in the dynamics of particles for the Stage 1 Lobatto integrator. The following notations are used in Eq. (66): q is the 3×1 vector of generalized coordinates to formalize the particle's translational positions in the absolute coordinate system, p is the 3×1 vector of generalized momenta, $V(\cdot)$ is a 3×1 vector of generalized velocities (as a function of generalized momenta), and $\left[G_{FF}^{abs}(\cdot) \right]_f$ is the 3×1 vector of generalized forces (external force applied to a particle and expressed in the absolute coordinate system).

In a more detail, the vector of generalized coordinates for a particle is the following:

$$q = r = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{67}$$

where the 3×1 vector r represents the position of a particle in the absolute Cartesian coordinate system.

Correspondingly, the vector of generalized momenta takes a form:

$$p = p_v = \begin{pmatrix} p_{v_x} \\ p_{v_y} \\ p_{v_z} \end{pmatrix} \tag{68}$$

where p_v is the 3×1momenta vector of a particle in the absolute coordinate system.

6.2.2 Dynamics of a Rigid Body

As was mentioned above, OMBI (namely, its rigid-body version considered in this document) is tailored for the specific dynamic equations expressed in the form:

$$\begin{aligned} \dot{q} &= V(q,p) \\ \dot{p} &= G(q,p) \end{aligned} \tag{69}$$

Eq. (69) formalizes dynamics of a rigid body in a multi-body system where bodies are not constrained but interact with each other (and with particles). The following notations are used in Eq. (69): q is the vector of generalized coordinates to formalize the body's translational and rotational positions in the absolute coordinate system, p is the vector of generalized momenta,

$V(\cdot)$ is a nonlinear vector-function formalizing kinematic relations (generalized velocities), and $G(\cdot)$ is the vector of generalized forces.

The following composition of the 7×1 vector of generalized coordinates, q, is used

$$q = \begin{pmatrix} r \\ \hline e \end{pmatrix} \tag{70}$$

where

$$r = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{71}$$

is the 3×1 vector of coordinates for the body-frame origin in the absolute coordinate system, and

$$e = \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix} \tag{72}$$

is the 4×1 vector of Euler parameters describing orientation of the body in the absolute coordinate system.

Note that the formulation of OMBI for rigid bodies requires that the body-frame origin is chosen in the body's COM (Center Of Mass) and the body-frame axes are the principal axes.

The following vector of generalized momenta is associated with the position vector of Eq. (70):

$$p = \begin{pmatrix} p_v \\ \hline p_\omega \end{pmatrix} \tag{73}$$

where

$$p_v = \begin{pmatrix} p_{v_x} \\ p_{v_y} \\ p_{v_z} \end{pmatrix} \tag{74}$$

is the 3×1 vector of translational momenta projected on the axes of the absolute coordinate system, and

$$p_\omega = \begin{pmatrix} p_{\omega_x} \\ p_{\omega_y} \\ p_{\omega_z} \end{pmatrix} \tag{75}$$

is the 3×1 vector of angular (rotational) momenta projected on the body-frame axes.

Note that the translational momenta are expressed in the absolute coordinate system while the rotational momenta are expressed in the body-frame. This is convenient in terms of providing a larger degree of separation between translational and rotational motions for rigid body (which, in its turn, helps to optimize the integrator).

Details of nonlinear functions $V(\cdot)$ and $G(\cdot)$ are omitted here. This document provides only the final integration algorithm, which integrates those functions in an explicit form.

6.2.3 *Inputs (for Particles and Bodies)*

q_0 is the vector of generalized coordinates at the beginning of the time step (marked by 0), size 7×1 (the first 3 components are Cartesian coordinates of the body's COM or of the particle and the second 4 components are Euler parameters).

Note: In the case of particle, the vector q_0 includes only the first 3 components.

p_0 is the vector of generalized momenta at the beginning of the time step (marked by 0), size 6×1 (the first 3 components are translational momenta and the second 3 components are rotational momenta).

Note: In the case of particle, the vector p_0 includes only the first 3 components.

Δt is the time step.

m is the mass of particle or body.

I_b is the vector of the body's principal inertia (I_x, I_y, I_z), size 3×1.

K_b is the vector of the body's inertia coefficients (K_x, K_y, K_z), size 3×1.

Note: The vector K_b is transformed from the vector I_b.

6.2.4 Outputs (for Particles and Bodies)

q_1 is the vector of generalized coordinates at the end of the time step (marked by 1), size 7×1 (same components as for q_0).

p_1 is the vector of generalized momenta at the end of the time step (marked by 1), size 6×1 (same components as for p_0).

6.2.5 External Operations (for Particles and Bodies)

$G_{FF}^{abs}(q)$

is a forcefield component of the generalized force (i.e. external force) in the absolute coordinate system as a function of position q, size 6×1 (the first 3 components are translational forces and the second 3 components are torques).

Note: In the case of particle, the vector G_{FF}^{abs} includes only the first 3 components.

6.2.6 Initialization Algorithm for OMBI

The initialization algorithm for OMBI is described separately for particles and bodies in order to highlight similarity of some operations and difference of other operations (when one works with a particle or a rigid body). The design of the OMBI initialization module in the OO C++ code should combine these operations when it is possible.

6.2.6.1 Initialization Algorithm for OMBI (Particles)

The following initialization operation is needed in order to start OMBI for particles. Note that these initialization operation is in addition to the initialization operations performed by the FORTRAN MBO(N)D code (MBO(N)D-I). This additional initialization operation is needed due to OMBI's specifics (momenta).

Initialize Translational Momenta for Particles

Transform the vector of initial translational velocities V into the vector of translational momenta p_v:

$$\begin{pmatrix} p_{v_x} \\ p_{v_y} \\ p_{v_z} \end{pmatrix} = \begin{pmatrix} m_p V_x \\ m_p V_y \\ m_p V_z \end{pmatrix} \tag{76}$$

where V_x, V_y, and V_z are projections of the vector of the particle's initial translational velocities V onto the axes of the absolute coordinate system, and m_p is the mass of the particle.

6.2.6.2 Initialization Algorithm for OMBI (Rigid Bodies)

The following initialization operations are needed in order to start OMBI for rigid bodies. Note that these initialization operations are in addition to the initialization operations performed by the FORTRAN MBO(N)D code. These additional initialization operations are needed due to OMBI's specifics: 1) the use of principal axes for setting the body-frame (instead of an arbitrary oriented body-frame); 2) the use of Euler parameters (instead of Euler angles); and, 3) the use of momenta (instead of velocities).

Additional Initialization Operations due to Principal Axes

Shift Body-Frame Origin to Body's COM

In the general MBO(N)D formulation the body-frame origin is normally not in the body's COM. So, in order to initialize OMBI, one needs to shift the body-frame origin to the COM. This is performed via the following sequence of operations.

1) Compute the body's COM (Center Of Mass) in the body-frame, Δr_{COM}:

$$\Delta r_{COM} = \frac{\sum_{j \in B} m_j^{atm} \left[\bar{r}_j^{nom} \right]^*}{m_b} \tag{77}$$

where $\left[\bar{r}_j^{nom} \right]^*$ is the vector of Cartesian coordinates of the j-th atom defined in the original body-frame (before its shift to COM), m_j^{atm} is the mass of the j-th atom, m_b is the mass of the body. Compute the new coordinates of atoms in the shifted body-frame:

$$\bar{r}_j^{nom} = \left[\bar{r}_j^{nom} \right]^* - \Delta r_{COM} \tag{78}$$

Note that these coordinates are needed for external operation to compute external force in OMBI.

Compute New Inertia Tensor (in the Shifted Body-Frame)

The new inertia tensor, $(I_b)^{**}$, is defined with respect to the body's COM and can be computed via the following sequence of operations.

1) Compute the shifted inertia, i.e. the inertia which are associated with the shift of the body-frame to the body's COM by the vector Δr_{COM}:

$$\begin{aligned}
\Delta I_{xx} &= m_b\left[\left(\Delta y_{COM}\right)^2 + \left(\Delta z_{COM}\right)^2\right] \\
\Delta I_{xy} &= m_b\left[\left(\Delta x_{COM}\right)\left(\Delta y_{COM}\right)\right] \\
\Delta I_{xz} &= m_b\left[\left(\Delta x_{COM}\right)\left(\Delta z_{COM}\right)\right] \\
\Delta I_{yy} &= m_b\left[\left(\Delta x_{COM}\right)^2 + \left(\Delta z_{COM}\right)^2\right] \\
\Delta I_{yz} &= m_b\left[\left(\Delta y_{COM}\right)\left(\Delta z_{COM}\right)\right] \\
\Delta I_{zz} &= m_b\left[\left(\Delta x_{COM}\right)^2 + \left(\Delta y_{COM}\right)^2\right]
\end{aligned} \tag{79}$$

where Δx_{COM}, Δy_{COM}, and Δz_{COM} are the components of the vector Δr_{COM}, and m_b is the mass of the body.

2) Assemble the shifted inertia tensor

$$\Delta I = \begin{pmatrix} \Delta I_{xx} & -\Delta I_{xy} & -\Delta I_{xz} \\ -\Delta I_{xy} & \Delta I_{yy} & -\Delta I_{yz} \\ -\Delta I_{xz} & -\Delta I_{yz} & \Delta I_{zz} \end{pmatrix} \tag{80}$$

3) Compute the new inertia tensor $(I_b)^{**}$:

$$[I_b]^{**} = [I_b]^{*} - \Delta I \tag{81}$$

Compute Principal Inertia and Rotation to Principal Axes

1) Perform eigenvalue decomposition of the non-diagonal inertia tensor (for each body):

$$(I_b)^{**} = \Lambda \cdot \mathrm{diag}(I_b)\Lambda^T \tag{82}$$

where $(I_b)^{**}$ is the non-diagonal 3×3 inertia tensor of the body, Λ is a 3×3 matrix whose columns are the corresponding eigenvectors, and $\mathrm{diag}(I_b)$ is a diagonal 3×3 matrix with the 3×1 vector of principal inertia $I_b = \left(I_x \;\; I_y \;\; I_z\right)^T$ on the diagonal. Note that the matrix Λ^T defines rotation from the original axes to the principal axes. The eigenvalue decomposition should be realized by a corresponding function from a C/C++ library.

2) Compute the new rotation matrix γ which rotates the absolute coordinate system to the new body-frame (with principal axes):

$$\gamma = \Lambda^T \gamma^* \tag{83}$$

where γ^* is the original 3×3 rotation matrix which rotates the absolute coordinate system to the original body-frame. The matrix γ^* is available in the FORTRAN MBO(N)D code (after the least squares fitting).

Compute Inertia Coefficients

Compute the body's inertia coefficients by transforming the principal inertia:

$$K_x = \frac{I_y - I_z}{I_z I_y},\ K_y = \frac{I_z - I_x}{I_x I_z},\ K_z = \frac{I_x - I_y}{I_y I_x} \tag{84}$$

Additional Initialization Operations due Euler Parameters

Initialize Euler Parameters

This initialization operation is based on the use of the rotation matrix γ rather than on the Euler angles θ. In this case, the operation becomes invariant to the type of Euler angles.

The initialization operation is based on the general-purpose algorithm for computing the Euler parameters from the rotation matrix. Note that this algorithm ensures that the vector of Euler parameters exactly satisfies the constraint $|e| = 1$ (to be maintained automatically by OMBI during the integration).

Additional Initialization Operations due to Momenta

Initialize Translational Momenta for Rigid Bodies

1) Project the vector of translational velocities U^* (expressed in the original body-frame) onto the axes of the absolute coordinate system:

$$V = (\gamma^*)^T U^* \tag{85}$$

Note that the 3×1 vector U^* and the 3×3 rotation matrix γ^* are available in MBO(N)D after least squares fitting.

2) Transform the vector of translational velocities V into the vector of translational momenta p_v:

$$\begin{pmatrix} p_{v_x} \\ p_{v_y} \\ p_{v_z} \end{pmatrix} = \begin{pmatrix} m_b V_x \\ m_b V_y \\ m_b V_z \end{pmatrix} \tag{86}$$

where V_x, V_y, and V_z are projections of the vector of translational velocities V onto the axes of the absolute coordinate system, and m_b is the mass of the body.

Initialize Rotational Momenta for Rigid Bodies

1) Project the vector of angular velocities ω^* (expressed in the original body-frame) onto the axes of the new body-frame (with principal axes):

$$\omega = \Lambda^T \omega^* \tag{87}$$

where the rotation 3×3 matrix Λ^T is computed via the eigenvalue decomposition and physically formalizes rotation from the original body-frame to the new body-frame (with principal axes).

2) Transform the vector of angular velocities ω into the vector of rotational momenta p_ω:

$$\begin{pmatrix} p_{\omega_x} \\ p_{\omega_y} \\ p_{\omega_z} \end{pmatrix} = \begin{pmatrix} I_x\omega_x \\ I_y\omega_y \\ I_z\omega_z \end{pmatrix} \tag{88}$$

where ω_x, ω_y, and ω_z are projections of the vector of angular velocities ω onto the axes of the new body-frame, and I_x, I_y, I_z are the principal inertia.

6.2.7 OMBI Algorithm

The OMBI algorithm for particles and bodies is described separately for particles and bodies in order to highlight similarity of some operations and difference of other operations for particles and rigid bodies. The design of the OMBI module in C++ should combine these operations when it is possible.

6.2.7.1 OMBI Algorithm (Particles)

The OMBI algorithm for particles is nothing else than a Verlet-type integrator or a second-order Runge-Kutta-Nystrom integrator. However, as was mentioned above, the OMBI algorithm for particles is described via a formalism of the OMBI algorithm for rigid bodies due to desired universality. The OMBI algorithm for particles can be formalized in the following block-scheme (see Figure 7).

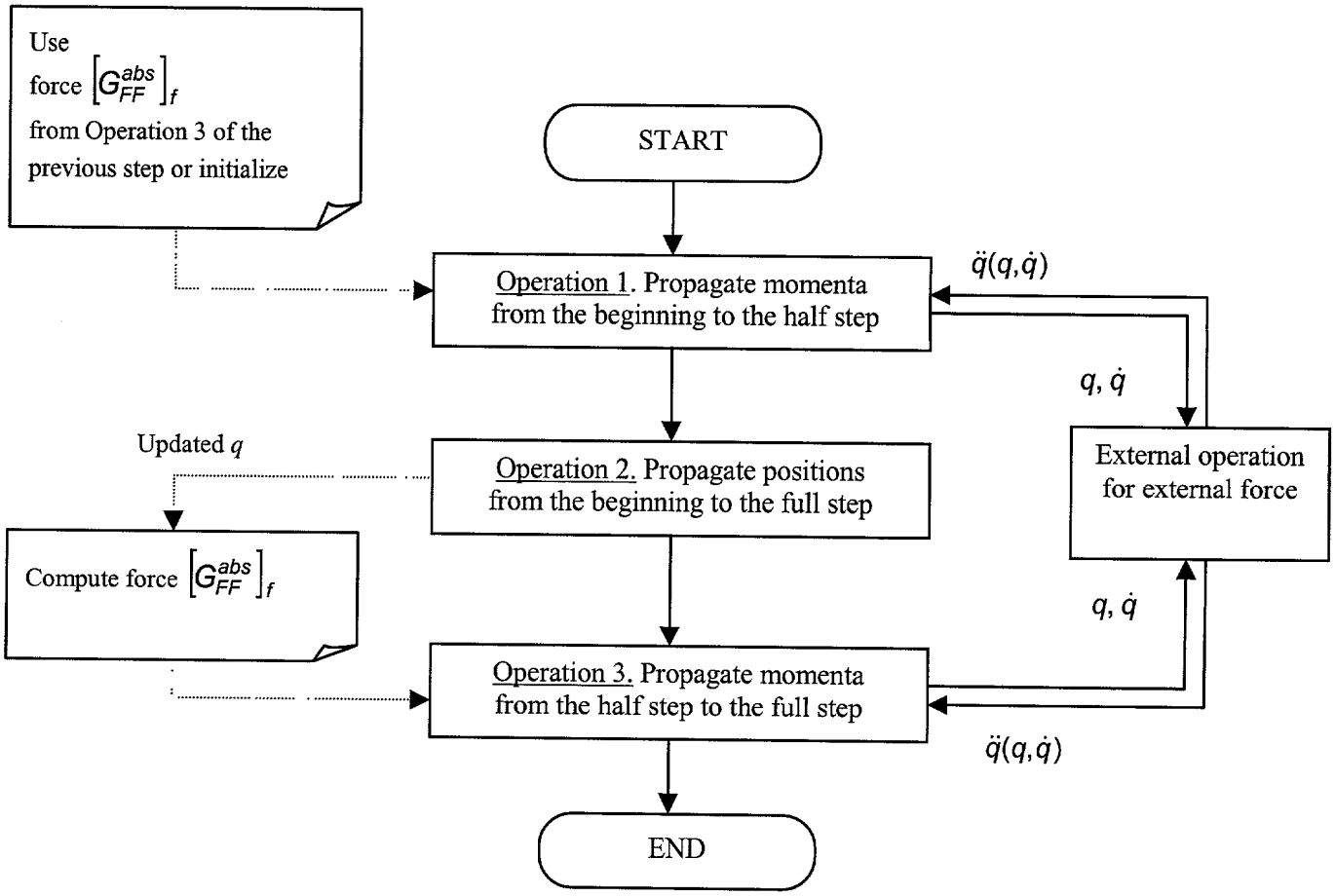


Figure 7. Block-Scheme of the OMBI Algorithm for Particles

Operation 1: Propagate Momenta from the Beginning of the Time Step to the Half Step

This operation includes propagating the 3×1 vector of translational momenta, p_v. It is organized via a function $P_v\{\cdot\}$. Note that introduction of this momenta propagation function is convenient for its reuse at Operation 3 of this OMBI for particles. Also, this function can be used for propagating the translational momenta of the body's COM in the OMBI for rigid bodies. Moreover, this function can be used in the next OMBI versions (e.g. in the Stage 3 OMBI for flexible bodies).

Compute Input for Function $P_v\{\cdot\}$

The input for function $P_v\{\cdot\}$ includes a position-dependent vector $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ which is the 3×1 translational force (the forcefield component of the generalized force) expressed in the absolute coordinate system.

As shown in Figure 7, the integration process should be organized in such a way that makes it possible to use (at Operation 1 of OMBI) the position-dependent external force $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ computed at Operation 3 of the previous time step.

Reusing the external force $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ is especially important due to the fact that evaluation of this force is very expensive in MD applications.

However, for the very first step of integration the external force $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ should be computed from initialization (as shown in Operation 3 of OMBI).

Propagate Translational Momenta via Function $P_v\{\cdot\}$

The vector of translational momenta is propagated from the beginning of the time step to the half step by using the following function

$$p_{v_{t/2}} = P_v\left\{\frac{\Delta t}{2},\ p_{v_0}\right\} \tag{89}$$

In Eq. (89), $P_v\{\cdot\}$ is a function defined as follows

$$p_v(t_s\ +\ \tau) = P_v\{\tau,\ p_v(t_s)\} \tag{90}$$

The following variables are the inputs for Eq. (90):

τ is the interval of propagation,

$p_v(t_s)$ is the 3×1 vector of translational momenta at the start point t_s,

$\left[G_{FF}^{abs}\right]_f$ is the 3×1 translational block (force) of the forcefield component of the generalized force in the absolute coordinate system (the first block in the vector G_{FF}^{abs}).

In Eq. (90) the output is the following:

$p_v(t_s\ +\ \tau)$ is the 3×1 vector of translational momenta at the finish point $t_s\ +\ \tau$.

Correspondingly, for Operation 1 the inputs are $\tau = \frac{\Delta t}{2}$, $p_v(t_s) = p_{v_0}$, and the output is $p_v(t_s + \tau) = p_{v_{1/2}}$ (where $t_s = 0$).

The function $P_v\{\cdot\}$ is computed as follows.

Propagate translational momenta in a vector form:

$$p_v(t_s + \tau) = p_v(t_s) + \left[G_{FF}^{abs}\right]_f \tau \tag{91}$$

Operation 2: Propagate Positions from the Beginning of the Time Step to the Full Step

This operation includes propagating the 3×1 vector of translational positions, r. It is organized via a function $Q_r\{\cdot\}$. Note that introduction of this position propagation function is convenient for its reuse in the case of OMBI (rigid bodies) for propagating the positions of the body's COM. Moreover, this function can be reused in next OMBI versions (e.g. in the Stage 3 OMBI for flexible bodies).

Propagate Translational Positions via Function $Q_r\{\cdot\}$

The vector of translational positions is propagated from the beginning of the time step to the full step by using the following function

$$r_1 = Q_r\left\{\Delta t,\ r_0,\ p_{v_{1/2}}\right\} \tag{92}$$

In Eq. (92), $Q_r\{\cdot\}$ is a function defined as follows

$$r(t_s + \tau) = Q_r\{\tau,\ r(t_s),\ p_v\} \tag{93}$$

The following variables are the inputs for Eq. (93):

τ is the interval of propagation,

$r(t_s)$ is the 3×1 vector of Cartesian coordinates at the start point t_s,

p_v is the 3×1 vector of translational momenta "frozen" at a particular instant,

m is the mass of the particle (or body).

In Eq. (93) the output is the following:

$r(t_s + \tau)$ is the 3×1 vector of Cartesian coordinates at the finish point $t_s + \tau$.

Correspondingly, for Operation 2 the inputs are $\tau = \Delta t$, $r(t_s) = r_0$, $p_v = p_{v_{1/2}}$ and the output is $r(t_s + \tau) = r_1$ (where $t_s = 0$).

The function $Q_r\{\cdot\}$ is computed via the following sequence of steps.

1) Convert the momentum vector p_v into the velocity vector v:

$$v = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} \frac{p_{v_x}}{m} \\ \frac{p_{v_y}}{m} \\ \frac{p_{v_z}}{m} \end{pmatrix} \tag{94}$$

2) Propagate the translational positions, r, from the beginning of the time step to the full step:

$$r(t_s + \tau) = r(t_s) + v\tau \tag{95}$$

Operation 3: Propagate Momenta from the Half Step to the Full Step

This operation includes propagating the vector of translational momenta, p_v. It is organized via a function $P_v\{\cdot\}$ (associated with Operation 1 of OMBI).

Compute Input for Function $P_v\{\cdot\}$

The input for function $P_v\{\cdot\}$ includes a position-dependent vector $\left\lfloor G_{FF}^{abs} \right\rfloor_f$.

The integration process should be organized in such a way that at Operation 3 of OMBI the position-dependent vector $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ should be recomputed. This due to the fact that the position vector q was changed at Operation 2 of OMBI.

Compute External Force $\left\lfloor G_{FF}^{abs} \right\rfloor_f$

The vector $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ of size 3×1 is a forcefield component of the generalized force applied to a particle and expressed in the absolute coordinate system.

The input $\left\lfloor G_{FF}^{abs} \right\rfloor_f$ is position-dependent and for Operation 3 it corresponds to the generalized positions at the end of the integration step: $q = q(\Delta t) = q_1$.

Propagate Translational Momenta

The vector of translational momenta is propagated from the half step to the full step by using the function $P_v\{\cdot\}$ (associated with Operation 1 of OMBI):

$$p_{v_1} = P_v\left\{\frac{\Delta t}{2},\ p_{v_{1/2}}\right\} \tag{96}$$

Correspondingly, as shown in Eq. (96) for Operation 3 the inputs to the function $P_v\{\cdot\}$ are $\tau = \frac{\Delta t}{2}$, $p_v(t_s) = p_{v_{1/2}}$, and the output is $p_v(t_s + \tau) = p_{v_1}$ (where $t_s = \frac{\Delta t}{2}$).

6.2.7.2 OMBI Algorithm (Rigid Bodies)

The OMBI algorithm for rigid bodies can be formalized in the following block-scheme (see Figure 8). Figure 8 is similar to Figure 7 (OMBI for particles) which is due to the fact that the

OMBI for particles is a particular case of the OMBI for rigid bodies. As shown in Figure 8, the OMBI algorithm for rigid bodies consists of three major operations.

Operation 1: Propagate Momenta from the Beginning of the Time Step to the Half Step

This operation includes propagating both translational p_v and rotational p_ω momenta. Correspondingly, the former is organized via a function $P_v\{\cdot\}$ and the latter is organazied via a function $P_\omega\{\cdot\}$. Note that the function $P_v\{\cdot\}$ can be reused from the OMBI for particles. Also, note that introduction of the rotational momenta propagation function $P_\omega\{\cdot\}$ is convenient for its reuse at Operation 3 of this OMBI and in the next OMBI versions (e.g. in the Stage 3 OMBI for flexible bodies).

Compute Inputs for Functions $P_v\{\cdot\}$ and $P_\omega\{\cdot\}$

The inputs for functions $P_v\{\cdot\}$ and $P_\omega\{\cdot\}$ include one position-dependent matrix and two position-dependent vectors: γ, $\left\lfloor G_{FF}^{abs} \right\rfloor_f$, and $\left\lfloor G_{FF}^{body} \right\rfloor_t$.

Correspondingly,

γ is the orthogonal rotation matrix of size 3×3 which rotates the absolute coordinate system to the body-frame,

$\left\lfloor G_{FF}^{abs} \right\rfloor_f$ is the 3×1 translational block (force) of the forcefield component of the generalized force in the absolute coordinate system (the first block in the vector G_{FF}^{abs}),

$\left\lfloor G_{FF}^{body} \right\rfloor_t$ is the 3×1 rotational block (torque) of the forcefield component of the generalized force in the body-frame (the second block in the vector G_{FF}^{body}).

As shown in Figure 8, the integration process should be organized in such a way that makes it possible to use (at Operation 1 of OMBI) the position-dependent matrices and vectors (γ, $\left\lfloor G_{FF}^{abs} \right\rfloor_f$, and $\left\lfloor G_{FF}^{body} \right\rfloor_t$) computed at Operation 3 of the previous time step.

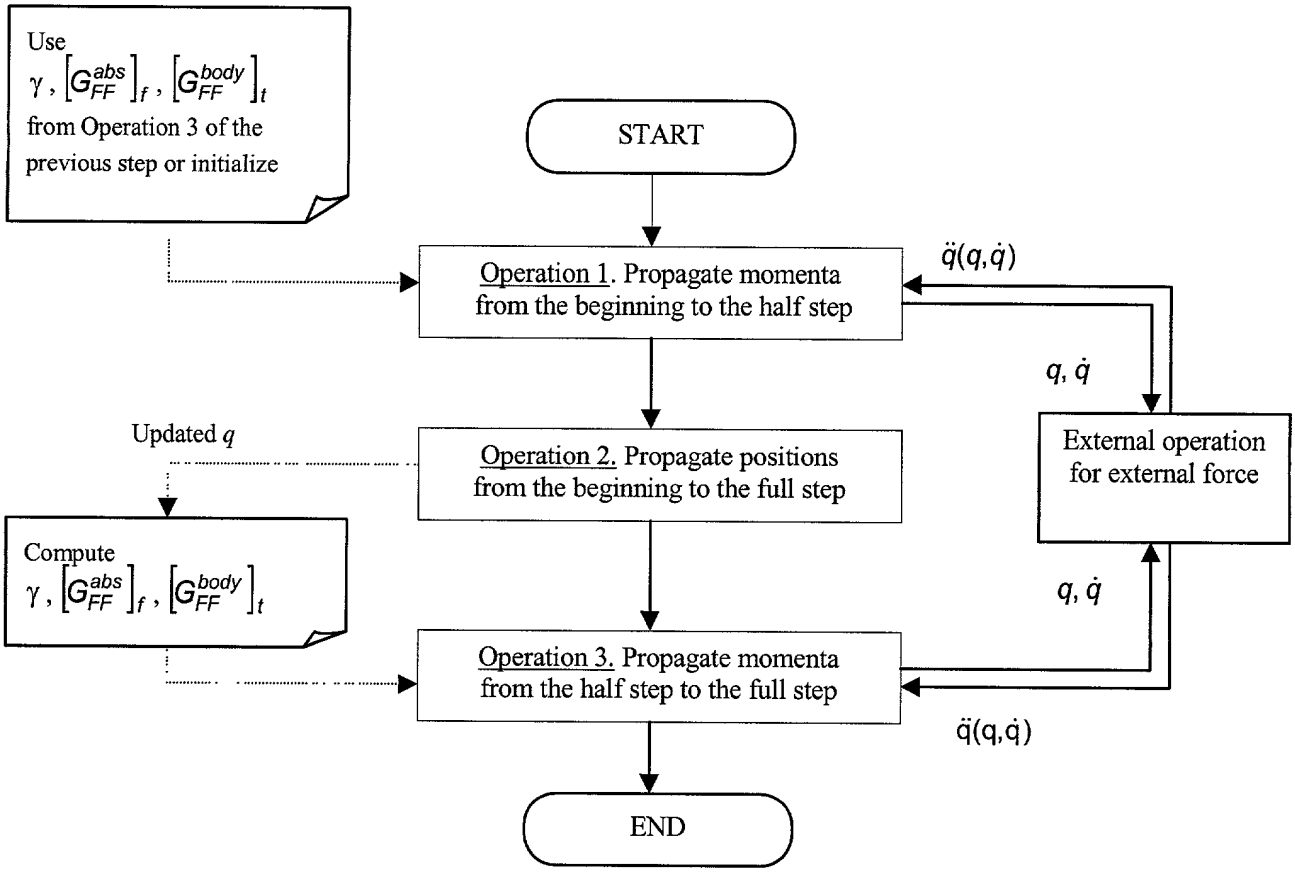


Figure 8. Block-Scheme of the OMBI Algorithm for Rigid Bodies

Reusing the external force G_{FF}^{abs} is especially important due to the fact that evaluation of this force is very expensive in MD applications.

However, for the very first step of integration the external force G_{FF}^{abs} and other position-dependent constructions (γ, etc.) should be computed from initialization (as shown in Operation 3 of OMBI).

Propagate Momenta via Functions $P_v\{\cdot\}$ and $P_\omega\{\cdot\}$

Propagate Translational Momenta via Function $P_v\{\cdot\}$

The vector of translational momenta is propagated from the beginning of the time step to the half step by using the following function

$$p_{v_{t/2}} = P_v\left\{\frac{\Delta t}{2},\ p_{v_0}\right\} \tag{97}$$

Note that the function $P_v\{\cdot\}$ can be reused from the OMBI for particles. Correspondingly, as shown in Eq. (97), for Operation 1 the inputs are $\tau = \frac{\Delta t}{2}$, $p_v(t_s) = p_{v_0}$, and the output is $p_v(t_s + \tau) = p_{v_{t/2}}$ (where $t_s = 0$).

Propagate Rotational Momenta via Function $P_\omega\{\cdot\}$

The vector of rotational momenta is propagated from the beginning of the time step to the half step by using the following function

$$p_{\omega_{1/2}} = P_\omega\left\{\frac{\Delta t}{2},\ p_{\omega_0}\right\} \tag{98}$$

In Eq. (98), $P_\omega\{\cdot\}$ is a function to be used twice in this OMBI (at Operations 1 and 3). It is defined as follows

$$p_\omega(t_s + \tau) = P_\omega\{\tau,\ p_\omega(t_s)\} \tag{99}$$

The following variables are the inputs for Eq. (99):

τ is the interval of propagation,

$p_\omega(t_s)$ is the 3×1 vector of rotational momenta at the start point t_s,

K_b is the 3×1 vector of the body's inertia coefficients (K_x, K_y, K_z),

γ is the orthogonal rotation matrix from the absolute coordinate system to the body-frame,

$\left[G_{FF}^{body}\right]_t$ is the 3×1 rotational block (torque) of the forcefield component of the generalized force in the body-frame.

In Eq. (99), the output is the following:

$p_\omega(t_s + \tau)$ is the 3×1 vector of rotational momenta at the finish point $t_s + \tau$.

Correspondingly, for Operation 1 the inputs are $\tau = \frac{\Delta t}{2}$, $p_\omega(t_s) = p_{\omega_0}$, and the output is $p_\omega(t_s + \tau) = p_{\omega_{1/2}}$ (where $t_s = 0$).

The function $P_\omega\{\cdot\}$ is computed as follows.

Propagate rotational momenta in a scalar form based on the Trotter decomposition:

$$\begin{aligned}
p_{\omega_x}(t_h) &= p_{\omega_x}(t_s) + \left\{K_x p_{\omega_y}(t_s) p_{\omega_z}(t_s) + \left[G_{FF}^{body}\right]_{t_x}\right\}\frac{\tau}{2}\\
p_{\omega_y}(t_h) &= p_{\omega_y}(t_s) + \left\{K_y p_{\omega_z}(t_s) p_{\omega_x}(t_h) + \left[G_{FF}^{body}\right]_{t_y}\right\}\frac{\tau}{2}\\
p_{\omega_z}(t_f) &= p_{\omega_z}(t_s) + \left\{K_z p_{\omega_x}(t_h) p_{\omega_y}(t_h) + \left[G_{FF}^{body}\right]_{t_z}\right\}\tau\\
p_{\omega_y}(t_f) &= p_{\omega_y}(t_h) + \left\{K_y p_{\omega_z}(t_f) p_{\omega_x}(t_h) + \left[G_{FF}^{body}\right]_{t_y}\right\}\frac{\tau}{2}\\
p_{\omega_x}(t_f) &= p_{\omega_x}(t_h) + \left\{K_x p_{\omega_y}(t_f) p_{\omega_z}(t_f) + \left[G_{FF}^{body}\right]_{t_x}\right\}\frac{\tau}{2}
\end{aligned} \tag{100}$$

In Eq. (100), the "half" ($t_h = t_s + \frac{\tau}{2}$) and "finish" ($t_f = t_s + \tau$) instants are introduced for briefness of notations (the values t_h and t_f do not need to be computed).

Operation 2: Propagate Positions from the Beginning of the Time Step to the Full Step

This operation includes propagating both translational r and rotational e positions. Correspondingly, the former is organized via a function $Q_r\{\cdot\}$ and the latter is organized via a function $Q_e\{\cdot\}$. Note that the function Q_r can be reused from the OMBI for particles. Also, note that introduction of the rotational position propagation function $Q_e\{\cdot\}$ is convenient for its reuse in the next OMBI versions (e.g. in the Stage 3 OMBI for flexible bodies).

Propagate Positions via Functions $Q_r\{\cdot\}$ and $Q_e\{\cdot\}$

Propagate Translational Positions via Function $Q_r\{\cdot\}$

The vector of translational positions is propagated from the beginning of the time step to the full step by using the following function

$$r_1 = Q_r\left\{\Delta t,\ r_0,\ p_{v_{1/2}}\right\} \tag{101}$$

Note that the function $Q_r\{\cdot\}$ can be reused from the OMBI for particles. Correspondingly, as shown in Eq. (101), for Operation 2 the inputs are $\tau = \Delta t$, $r(t_s) = r_0$, $p_v = p_{v_{1/2}}$ and the output is $r(t_s + \tau) = r_1$ (where $t_s = 0$).

Propagate Rotational Positions via Function $Q_e\{\cdot\}$

This operation should be organized in a separate reusable function since it can be used in different OMBI algorithms. In the case of OMBI for rigid bodies the operation is used only once.

The rotational positions are represented by the vector of Euler parameters e and are propagated as follows (at the full time step for rigid bodies)

$$e_1 = Q_e\left\{\Delta t,\ e_0,\ p_{\omega_{1/2}}\right\} \tag{102}$$

In Eq. (102), $Q_e\{\cdot\}$ is a function to be used in different OMBI algorithms. It is defined as follows

$$e_1 = Q_e\left\{\tau,\ e(t_s),\ p_\omega\right\} \tag{103}$$

The following variables are the inputs for Eq. (103):

τ is the interval of propagation,

$e(t_s)$ is the 4×1 vector of Euler parameters at the start point t_s,

p_ω is the 3×1 vector of angular momenta "frozen" at a particular instant.

I_b is the 3×1 vector of the body's principal inertia (I_x, I_y, I_z),

In Eq. (103), the output is the following:

$e(t_s + \tau)$ is the 4×1 vector of Euler parameters at the finish point $t_s + \tau$.

Correspondingly, for Operation 2 the inputs are $\tau = \Delta t$, $e(t_s) = e_0$, $p_\omega = p_{\omega_{1/2}}$ and the output is $e(t_s + \tau) = e_1$ (where $t_s = 0$).

The function $Q_e\{\cdot\}$ is computed via the following sequence of steps.

1) Convert the angular momentum vector p_ω to the angular velocity vector ω (both expressed in the body-frame):

$$\omega = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} \dfrac{p_{\omega_x}}{I_x} \\ \dfrac{p_{\omega_y}}{I_y} \\ \dfrac{p_{\omega_z}}{I_z} \end{pmatrix} \tag{104}$$

2) Calculate the magnitude of the angular velocity vector ω:

$$|\omega| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{105}$$

3) Calculate the half of the magnitude $|\omega|$:

$$\nu = \frac{|\omega|}{2} \tag{106}$$

4) Precalculate the following sine and cosine:

$$c_\nu = \cos(\nu\tau),\ s_\nu = \sin(\nu\tau) \tag{107}$$

5) Calculate the normalized angular velocity vector in the body-frame, $\bar{\omega}$:

$$\begin{pmatrix} \bar{\omega}_x \\ \bar{\omega}_y \\ \bar{\omega}_z \end{pmatrix} = \begin{pmatrix} \dfrac{\omega_x}{|\omega|} \\ \dfrac{\omega_y}{|\omega|} \\ \dfrac{\omega_z}{|\omega|} \end{pmatrix} \tag{108}$$

6) Evaluate analytically the matrix exponent $\Psi(\omega)$ (note that only the elements in the upper triangular are shown due to symmetry):

$$\Psi(\omega) = \begin{pmatrix} c_\nu & s_\nu\bar{\omega}_x & s_\nu\bar{\omega}_y & s_\nu\bar{\omega}_z \\ & c_\nu & s_\nu\bar{\omega}_z & s_\nu\bar{\omega}_y \\ & & c_\nu & s_\nu\bar{\omega}_x \\ & & & c_\nu \end{pmatrix} \tag{109}$$

7) Propagate the vector of Euler parameters over the specified interval τ:

$$e(t_s + \tau) = \Psi(\omega)e(t_s) \tag{110}$$

Operation 3: Propagate Momenta from the Half Step to the Full Step

This operation includes propagating both translational p_ν and rotational p_ω momenta. Correspondingly, the former is organized via a function $P_\nu\{\cdot\}$ and the latter is organized via a function $P_\omega\{\cdot\}$ (associated with Operation 1 of OMBI).

Compute Inputs for Functions $P_v\{\cdot\}$ and $P_\omega\{\cdot\}$

The inputs for functions $P_v\{\cdot\}$ and $P_\omega\{\cdot\}$ include one position-dependent matrix and two position-dependent vectors: γ, $\left[G_{FF}^{abs}\right]_f$, and $\left[G_{FF}^{body}\right]_t$. These matrix and vectors are described for Operation 1 of OMBI.

The integration process should be organized in such a way that at Operation 3 of OMBI the position-dependent matrices and vectors (γ, $\left[G_{FF}^{abs}\right]_f$, and $\left[G_{FF}^{body}\right]_t$) should be recomputed. This due to the fact that the position vector q was changed at Operation 2 of OMBI.

Compute Rotation Matrix γ

The orthogonal rotation matrix γ of size 3×3 rotates the absolute coordinate system to the body-frame. This matrix is needed only for rigid bodies (not particles).

The rotation matrix γ is computed as a function of the four Euler parameters e. Note that for Operation 3 the vector of Euler parameters corresponds to the end of the integration step: $e = e(\Delta t) = e_1$. Also note that the vector of Euler parameters e of size 4×1 is the first block in the vector of generalized positions q.

Compute External Force G_{FF}^{abs}

The vector G_{FF}^{abs} of size 6×1 is a forcefield component of the generalized force expressed in the absolute coordinate system. The vector G_{FF}^{abs} includes two blocks: translational force and torque. The translational force, $\left[G_{FF}^{abs}\right]_f$, is a vector of size 3×1 representing the force applied to the body. The torque, $\left[G_{FF}^{abs}\right]_t$, is the vector of size 3×1 representing the torque applied about the body-frame origin. Both are expressed in the absolute coordinate system.

The input G_{FF}^{abs} is position-dependent and for Operation 3 it corresponds to the generalized positions at the end of the integration step: $q = q(\Delta t) = q_1$.

Compute Torque $\left[G_{FF}^{body}\right]_t$ in Body-Frame

Project the torque vector from the absolute coordinate system to the body-frame:

$$\left[G_{FF}^{body}\right]_t = \gamma\left[G_{FF}^{abs}\right]_t \tag{111}$$

where γ is the rotation matrix.

Propagate Momenta via Functions $P_v\{\cdot\}$ and $P_\omega\{\cdot\}$

Propagate Translational Momenta via Function $P_v\{\cdot\}$

The vector of translational momenta is propagated from the half step to the full step by using the function $P_v\{\cdot\}$ (associated with Operation 1 of OMBI):

$$p_{v_1} = P_v\left\{\frac{\Delta t}{2},\ p_{v_{1/2}}\right\} \tag{112}$$

Correspondingly, as shown in Eq. (112), for Operation 3 the inputs to the function $P_v\{\cdot\}$ are $\tau = \frac{\Delta t}{2}$, $p_v(t_s) = p_{v_{1/2}}$, and the output is $p_v(t_s + \tau) = p_{v_1}$ (where $t_s = \frac{\Delta t}{2}$).

Propagate Rotational Momenta via Function $P_\omega\{\cdot\}$

The vector of rotational momenta is propagated from the half step to the full step by using the function $P_\omega\{\cdot\}$ (associated with Operation 1 of OMBI):

$$p_{\omega_1} = P_\omega\left\{\frac{\Delta t}{2},\ p_{\omega_{1/2}}\right\} \tag{113}$$

Correspondingly, as shown in Eq. (113), for Operation 3 the inputs to the function $P_\omega\{\cdot\}$ are $\tau = \frac{\Delta t}{2}$, $p_\omega(t_s) = p_{\omega_{1/2}}$, and the output is $p_\omega(t_s + \tau) = p_{\omega_1}$ (where $t_s = \frac{\Delta t}{2}$).

7 Task 3: Implement Efficient Constraint-Handling Formulation/Architecture

In the previous Sections the OMBI algorithm was introduced for unconstrained multibody dynamics. In this Section we present the OMBI in a combination with SHAKE formulation to account for the bond-length constraints (Ref. 51). It should be noted that this formulation for the constrained multi-body dynamics was chosen during the Phase II work from several candidate methodologies under consideration for treating constrained MBD systems with the OMBI (as described below).

Enforcing constraints is a well-proven technique used in molecular dynamics for atomistic models realized in such algorithms as SHAKE, RATTLE, etc. (Refs. 34, 51). In the body-based formulation, substructuring of atoms into bodies already means introducing of "hard" (if a body is rigid) or "soft" (if a body is flexible) constraints, and, thus, reducing the high-frequency dynamics. However, enforcing constraints between bodies helps to make further improvements in extracting the essential (low-frequency) dynamics from less interesting high-frequency motions (e.g., bond-stretching motion). The MBO(N)D technology offers an efficient O(N) algorithm for modeling dynamics of a system of connected bodies (Refs. 60, 61). This algorithm is based on the use of relative hinge coordinates and makes it possible to enforce *exact* constraints on the parameters of interest such as bond lengths, bond angles, and dihedral angles while integrating motion with respect to the remaining "free" coordinates (unconstrained degrees of freedom). Our experience in simulating reduced-variable constrained molecular dynamics by MBO(N)D indicates that enforcing bond length constraints between bodies will usually suffice for the goal of removing high-frequency dynamics. Therefore, the task of incorporating constraints into OMBI is reduced to that of accounting for bond length constraints.

The above simplification (enforcing only bond length constraints), combined with the re-formulation of the multibody dynamics in terms of momenta and Euler parameters, provided a unique opportunity for incorporating constraints. We therefore decided to take a broader look at this problem and to investigate three possible approaches: 1) $O[(n+m)]^3$ SHAKE-type solution (Ref. 51) which is based on Gauss-Seidel iterations and modified by us for body-based formulation, 2) $O(n+m)$ recursive algorithm which is the core algorithm for constrained multibody dynamics in MBO(N)D, and 3) the impetus-striction method which is based on a reformulation of the constrained Lagrangian systems in which constraints are manifested as integrals of motion (Refs. 19-23). With the proven (during the Phase I work) feasibility of the OMBI for unconstrained dynamics in absolute coordinates as well as the fact that all three candidate approaches mentioned above for incorporating constraints are proven concepts which have been

in use for a long time, it was clear that the task of generalizing the OMBI to the case of constrained dynamics was a low risk task which, however, helped to significantly enhance the performance of MBO(N)D-II. As was mentioned above, in MBO(N)D-II we accounted for the constraints via the OMBI in a combination with SHAKE formulation.

7.1 Candidate Approaches to Constrained Dynamics

In Phase II Proposal for this research work, three possible approaches for constrained multibody dynamics were proposed, namely: 1) SHAKE-like algorithm for body-based formulation, 2) O(n+m) recursive algorithm, and 3) Impetus-striction method. During the Phase II work we investigated all these three approaches and decided to implement in the OO C++ code only the first (SHAKE-like) approach. Correspondingly, we finalized the development and carried out implementation of the combined OMBI/SHAKE algorithm, which accounts for the bond-length constraints between bodies and particles.

In this Section we still briefly describe the O(n+m) recursive algorithm, and the Impetus-striction method which were investigated but not selected for their implementation in MBO(N)D-II.

7.1.1 O(n+m) Recursive Algorithm

Another alternative in the development of the OMBI for constrained molecular dynamics is the incorporation of the *O(n+m)* recursive algorithm used in the current version of MBO(N)D (but, with the Euler angles parametrization). The *O(n+m)* recursive algorithm offers a more elegant way of solving a system of constrained equations than SHAKE-like iterations for tree topologies. The advantage of this solution is that no SHAKE-like iterations are required since the Lagrange multipliers are explicitly eliminated from the equations of motion due to the use of relative (hinge) coordinates. The current *O(n+m)* algorithm in MBO(N)D consists of three recursive sweeps through the system topology - a first forward recursion to obtain the absolute velocities from the relative velocities, a backward recursion to compute equivalent inertias and forces, and a second forward recursion to evaluate the relative accelerations. For closed-loop topologies, a hinge is arbitrarily selected for treatment as a *cut-joint*, whereby the local topology becomes a tree if all constraints are removed from the cut-joint hinge. Upon selection of appropriate cut-joints, the system can then be treated as a topological tree, subject to constraint forces that are present at the cut-joints. Additional recursive calculations are done to explicitly solve for the cut-joint Lagrange multipliers that enforce those constraints.

No major changes are needed in the *O(n+m)* algorithm itself for incorporation into OMBI. The minor changes that are needed involve new kinematics in the form of Euler parameters instead of Euler angles, and new dynamics terms in the form of momenta instead of velocities. Since the O(n+m) algorithm is the most efficient for tree topologies, it makes sense to only apply the SHAKE-like iterations for the constraints related to closed-loop topologies, where Lagrange multipliers are calculated enforcing constraints at the cut joints. Since there is usually a small number of closed-loops in most biological molecules (and substructuring makes this number even smaller), the system that the iterative solution is applied to will be of low dimension, while most of the hinge constraints between bodies (associated with the topological tree portions of the model, and hence treated with the *O(n+m)* algorithm) will be eliminated explicitly.

The fact that we needed to treat the closed-loops through the SHAKE-type algorithm, finally lead us to the conclusion that open-loop constraints should be also treated through the SHAKE-type algorithm. The hybrid SHAKE/O(N) formulation would have introduced unnecessary complexity in the OO C++ MBO(N)D-II code. At the same time, the universal treatment of the open- and closed-loop topologies in molecular systems allowed for a more efficient treatment of large-scale MD simulations with a bulk of explicit water molecules (modeled

via unconstrained multi-body dynamics which were integrated via the OMBI described in Section 6).

7.1.2 Impetus-Striction Method

A novel approach to constrained MD involves a new mathematical formulation of holonomically constrained systems called the impetus-striction approach. In Ref. 19 it is shown that infinite dimensional (i.e., PDE) constrained Lagrangian dynamical systems can effectively be written as unconstrained Hamiltonian systems in which the constraints are, by construction, integrals of the motion. These ideas have already proven useful in various macroscopic examples of classical continuum mechanics. In the finite dimensional (i.e., ODE) context of constrained MD the new technique is a simple variant of Kozlov's "vakonomic" mechanics (Ref. 19). However even in the ODE context the numerical ramifications of the new formulation do not appear to have been explored. In particular we continue to investigate whether the new Hamiltonian systems will yield more efficient schemes for MD simulations. For example in MD with quadratic bond length (and angle) constraints the differential algebraic problem of integrating constrained second-order time dynamics can be recast as the conservative integration of Hamiltonian dynamics with quadratic integrals.

The idea of the impetus-striction formulation is as follows. Many systems can naturally be formulated as Lagrangian systems that are subject to constraints. The bond length and angle constraints in molecular dynamics are certainly of this type, as are the multibody constraints that arise in MBO(N)D. Consideration of the dynamics of inextensible and unshearable rods (Refs. 19-23) led to a novel *unconstrained* Hamiltonian formulation of a rather general class of such Lagrangian systems. The desired constraint is by construction a first integral of the Hamiltonian dynamics. However the Hamiltonian, $H(x,y)$ say, is only defined after an auxiliary minimization

$$H(x,y) = \min_{\lambda} \tilde{H}(x,y,\lambda) \tag{114}$$

where we call tilde $H(x,y,\lambda)$ the *pre-Hamiltonian*. Here y is the conjugate variable to the configuration variable x, but in applications y is typically not the classic momentum or impulse, and so we give it a new name, the *impetus*. In applications the quantity λ is generally the time anti-derivative of a familiar physical quantity, e.g., for incompressible fluid flow λ_t is the pressure field. λ can also be interpreted as a Lagrange multiplier enforcing a time-differentiated constraint, but to distinguish it from the usual multiplier, we call λ the *striction*.

The new formulation holds great promise in the context of multibody molecular dynamics. Moreover, due to fact that the implementation of the impetus-striction method to the quadratic (bond length) constraints is straightforward, we considered this task as a low-risk task. The main effort is undertaken in order to merge the impetus-striction methodology of handling constraints with the explicit integrator for multibody dynamics, namely the OMBI. It should be mentioned that at this point in time, the impetus-striction method is used with the implicit symplectic Runge-Kutta discretizations since they automatically conserve quadratic integrals. However, in MD the use of implicit integrators is prohibitive due to large computational cost of forcefield evaluations.

The fact that it was practically impossible to implement the impetus-striction formalism in a form of the explicit integrator, lead us to the decision to abandon this approach and finalize the MBO(N)D-II developments with the SHAKE-type constrained formulation. The practical difficulty in implementing the impetus-striction formalism consisted in the fact that we discovered that the implicit integrator was needed to ensure the stable integration process of the constrained multi-body dynamics under conditions that the constraints are maintained only at the velocity level. A combination of the explicit integrator and the treatment of the constraints at the velocity level appeared to be insufficient for the stable integration. Correspondingly, we concluded that a

combination of the explicit integrator (OMBI) with the SHAKE-type formalism (which treats constraints at the position level) is a more powerful approach to the constrained multi-body MD.

7.2 SHAKE-like Algorithm for Body-Based Formulation

The SHAKE-like approach offers a very simple and straightforward way of enforcing bond-length constraints, which has been proven to be a way of retaining the symplectic property of the Verlet-type integrator for dynamical systems with velocity-independent terms (e.g., atomistic molecular systems, Ref. 51). The main idea consists in the iterative Gauss-Seidel-Newton (GSN) solution of a system of nonlinear equations which represent discrepancies between the bond lengths to be maintained and the bond lengths predicted at the end of the integration step. The GSN iterations may include a successive overrelaxation (SOR) scheme to increase the speed of convergence. In the nonlinear equations that arise, the prediction of the momentum and position vector within the integration step are performed in the same way as in the Verlet-type integrator, but with additional constraint forces characterized by Lagrange multipliers.

The OMBI developed on the basis of the Liouville-Trotter formalism is a generalization of the Verlet-type integrator in the sense that the Liouville-Trotter decomposition does not provide the final finite-difference equation to propagate the momentum or position states but only decomposes the system of differential equations into a system of simpler differential equations (see Section 6.1). In order to propagate momentum or position states within the integration step, one has to solve a system of differential equations. In Section 6.1 we developed analytical solutions of the systems of differential equations arising as a result of the Liouville-Trotter decomposition of equations of multibody dynamics for the formulation with Euler parameters and momenta. These results can be extended to the case of constrained dynamics and the formalism of the SHAKE method can be directly used for generalizing OMBI to the case of constrained dynamics. Moreover, since OMBI demonstrates conservation properties which indicate that it is most likely a symplectic integrator (we plan to prove this mathematically), the SHAKE-type iterations will not degrade those properties, i.e., it is expected that the OMBI for constrained dynamics will retain its high performance in terms of conserving the total energy and momenta in the conservative system. It should also be mentioned that the SHAKE-like approach has such advantageous properties as handling both open- and closed-loop topologies of the linked bodies, and is very amenable for parallelization.

In section below we explicitly formulate a SHAKE-type algorithm for use with the OMBI.

7.3 Description of the OMBI for the General Case of a System of Constrained Flexible Bodies and Particles – OMBI/SHAKE

In this Section we will present the results, which generalize the OMBI from the case of the unconstrained flexible bodies to the case of a system of constrained flexible bodies and particles.

In atomistic molecular dynamics, the classical combination of the Verlet integrator and SHAKE algorithm (Ref. 48 and 53) is a well-proven technique to increase the integration step (by 2-4 times) by removing rapid vibrational modes associated with the bonds. The combination of a SHAKE-like approach with the body-based OMBI algorithm is a direct generalization of the above classical results to the case when a molecule is substructured into a set of flexible bodies and particles. It will become clear below that this generalization is not trivial.

7.3.1 Formulation of Constrained Dynamics for Multi-Body System

In Section 6.1 the formulation of multibody dynamics in terms of Euler parameters was described for unconstrained bodies. This fact made it possible to simplify the integrator's derivations by considering each body in the system individually. This is not the case for interconnected bodies. While deriving the OMBI for the case of constrained dynamics, we will use the following notation. The multibody dynamics (yet to be constrained) will be described exactly in the form of Eq. (22) (for a single body):

$$\begin{aligned} \dot{q} &= V(q,p) \\ \dot{p} &= G(q,p) \end{aligned} \tag{115}$$

But the state-vector q and the momentum vector p will have the block-vector structure:

$$q = \begin{pmatrix} q_1 \\ \hline \vdots \\ \hline q_N \end{pmatrix}, \; p = \begin{pmatrix} p_1 \\ \hline \vdots \\ \hline p_N \end{pmatrix} \tag{116}$$

where each i-th block corresponds to the i-th body in the system of N bodies.

Moreover, we will assume that the system may include both flexible bodies and particles. In the case when the i-th block corresponds to a flexible body, the position vector for the body includes 3 sub-blocks: e - the 4×1 vector of Euler parameters, x - the 3×1 vector of translational coordinates of the body's center of mass (COM), and ξ - the vector $m_i\times1$ of modal coordinates (see Eq. (24)). The momentum vector for each body also includes 3 sub-blocks: p_ω – (3×1) , p_u – (3×1) , and p_ξ – $(m_i\times1)$. The momentum vector is introduced as a linear combination of the corresponding velocity vector (see Eq. (26)) which includes the 3×1 block ω of the body's angular velocities projected onto the body frame, the the 3×1 block u of the translational velocities of the body's COM projected onto the body frame, and the $m_i\times1$ block of the modal velocities. A general form of the mass matrix for each flexible body is introduced in Eq. (27). Note that the case of a rigid body is a particular case of a flexible body if one excludes the ξ-blocks from the corresponding position and momentum vectors as well as from the mass matrix. Similarly, in the case when the i-th block in the system corresponds to a particle, the position vector for the body includes only 3 translational coordinates x, and the momentum vector includes only 3 elements expressed through translational velocities u (projected on to absolute axes). In this case the mass matrix is a 3×3 diagonal matrix with the particle's mass on the diagonal.

Let us introduce bond-length constraints into the multibody dynamics, i.e. assume that the bodies in the system are interconnected by means of bonds. Normally, two bodies are connected by only one bond, which links two specified atoms on both bodies. However, the molecular structure may consist of both open- or closed-loop topologies. We assume that there are L bond-length constraints in the system which can be formalized by one vector equation

$$g(q) = 0 \tag{117}$$

Note that Eq. (117) falls into the class of holonomic (position) constraints. Also, note that the constraint is expressed in the most general form as a function of the full state-vector q. But, only the i-th and j-th blocks of the state-vector q are directly involved in forming a bond-length constraint between i-th and j-th bodies in the system:

$$g(q_i, q_j) = 0 \tag{118}$$

This determines the structure of the sparsity in the corresponding matrix constructions that will arise in the process of deriving the equations for constrained multibody dynamics. In a more detailed form each *l*-th constraint may be expressed as

$$\left[X_{ik}(q_i) - X_{jr}(q_j)\right]^2 + \left[Y_{ik}(q_i) - Y_{jr}(q_j)\right]^2 + \left[Z_{ik}(q_i) - Z_{jr}(q_j)\right]^2 - b_l^2 = 0 \qquad (119)$$

where the triple $\{X, Y, Z\}$ stands for the absolute Cartesian coordinates of the atom participating in the bond-length constraint. The index *i* or *j* denotes the body's number in the system, while the index *k* or *r* denotes the atom's index within the body. Further specification of the form of Eq. (119) detailization depends on whether the atom participating in the bond is an individual particle or belongs to a rigid or flexible body.

In the case of a particle, its three Cartesian coordinates coincide with the corresponding block of the state-vector:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = q_i \quad (k = 1) \qquad (120)$$

In the case of a body, the 3 Cartesian coordinates of the *k*-th atom in the *i*-th body are defined by the following transformational equation:

$$\begin{pmatrix} X_{ik} \\ Y_{ik} \\ Z_{ik} \end{pmatrix} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}_{COM} + C(e_i) \begin{pmatrix} \Delta X_{ik} \\ \Delta Y_{ik} \\ \Delta Z_{ik} \end{pmatrix} \qquad (121)$$

where the first term in the right-hand side represents the absolute position of the COM for the i-th body and the second term represents relative coordinates (marked with Δ) of the k-th atom in the body frame rotated in absolute space according to the rotation matrix $C(e_i)$. Note that the matrix $C(e_i)$ depends quadratically on the four Euler parameters (see Eq. (32)). In the general case of a flexible body, the vector of relative coordinates for the *k*-th atom in the body frame depends on the modal states which model deformational motion:

$$\begin{pmatrix} \Delta X_{ik} \\ \Delta Y_{ik} \\ \Delta Z_{ik} \end{pmatrix} = \begin{pmatrix} \Delta X_{ik} \\ \Delta Y_{ik} \\ \Delta Z_{ik} \end{pmatrix}_0 + \Phi \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_{m_i} \end{pmatrix} \qquad (122)$$

Here the index "0" corresponds to the non-deformed state of the body, and the $(3 \times m_i)$ matrix Φ is a matrix of the corresponding mode shapes which project the current modal states into the current relative coordinates of the *k*-th atom in the body frame. Note that the case of rigid body entails that $\xi \equiv 0$.

Eqs. (119)-(122) applied to each bond constitute a general nonlinear vector constraint of Eq. (117). The structure of this vector constraint is defined by the connectivities in the system and by the type of body (particle, rigid body, flexible body). It should be emphasized that the above model of bond-length constraints for a set of flexible bodies has a convenient analytical representation which involves only polynomial nonlinearities. This is achieved through the use of an Euler parameter formulation for the description of the rotational motion of bodies. The convenient (polynomial) form of the constraint equations is a crucial factor in merging the OMBI with the SHAKE-type algorithm for constrained dynamics.

According to a variational principle (see e.g. Ref. 11), the equations of motion (115) subject to L holonomic constraints (117) may be written in the form:

$$\begin{aligned} \dot{q} &= V(q,p) \\ \dot{p} &= G(q,p) + [g'(q)]^T \lambda \\ 0 &= g(q) \end{aligned} \tag{123}$$

where λ is a $(L\times 1)$ vector of time-dependent Lagrange multipliers (constraint forces) and $g'(q)$ is $(L\times n)$ Jacobian matrix for the vector constraint of Eq. (117):

$$g'(q) = \frac{\partial}{\partial q} g(q) \tag{124}$$

Note that the Jacobian matrix is very sparse due to the aforementioned topology of connectivities. Indeed, only two bodies are connected by a bond. Moreover, for protein molecules, for example, the topological structure is mostly open (tree topology) with a few possible closed loops. Substructuring molecules into bodies tends to eliminate most of the closed loops (such as rings) by including them within a single (rigid or flexible) body.

Eq. (123) forms a system of differential-algebraic equations (DAEs) of index three; three differentiations are required in order to reduce the equations to a system of ordinary differential equations (see, e.g., Ref. 11). The solution manifold underlying Eqs. (123), and (124) is $\mathrm{M} = \left\{ (q,\ p) \;\middle|\; g(q) = 0,\ g'(q)DM^{-1}p = 0 \right\}$. Note that here the "hidden" constraint $g'(q)DM^{-1}p = 0$ is obtained through time differentiation of the position constraint and is nothing more than the original bond-length constraint at the velocity level.

7.3.2 Liouville-Trotter Formalism for Decomposition of Constrained Dynamics

While deriving the OMBI for unconstrained multibody dynamics (see Section 6.1) it was emphasized that the Liouville-Trotter formalism leads to an efficient time-symmetric decomposition of the original system of differential equations into a set of simpler systems of differential equations. This principally differs from the case of atomistic molecular dynamics where the Liouville-Trotter formalism directly yields a time-symmetric discretization of the system of differential equations in the form of the Verlet integrator (RESPA-type approach, see Ref. 45). Correspondingly, the art of designing the OMBI consisted in solving the decomposed equations while taking full advantage of their analytical properties. The same approach will be implemented for constrained dynamics in order to derive the OMBI/SHAKE algorithm.

Using the Liouville-Trotter formalism, we decomposed the system of Eq. (123) into five subsystems of differential equations (in Section 6.1.3 for unconstrained multibody dynamics). In the case of constrained dynamics, the solutions of those subsystems are coupled by the need to satisfy additional algebraic conditions, namely $g(q) = 0$ and $g'(q)DM^{-1}p = 0$ (i.e., to satisfy the bond-length constraints at on the position and velocity levels).

The decomposed equations of constrained motion include five stages presented below as a generalization of the five-stage OMBI for unconstrained dynamics developed in Section 6.1.3.

Stage 1. Propagate the momentum vector to the half step $t = \Delta t/2$ freezing the position states and the vector of Lagrange multipliers at $t = 0$:

$$\dot{p} = G(q_0,p) + [g'(q_0)]^T \lambda_0,\ t \in \left(0, \frac{\Delta t}{2}\right) \tag{125}$$

with initial conditions $p(0) = p_0$. Note that unlike the case of unconstrained multibody dynamics (see Eq. (38)), now the "semi-frozen" equation for momenta is no longer a closed expression for $p_{1/2}$ since one has to evaluate the vector of Lagrange multipliers λ_0 from the condition of satisfying the bond-length constraints on the position level:

$$g(q_1) = 0 \tag{126}$$

where vector q_1 is the state-vector of the system at the end of the time step $t = \Delta t$ and is available only after the next 3 stages are realized.

Stage 2. Propagate the vector of rigid position variables to the half step $t = \Delta t/2$ freezing the flexible position variables at $t = 0$ and the momentum vector at $t = \Delta t/2$:

$$\dot{z} = V_z\left(z, \xi_0, p_{1/2}(\lambda_0)\right), \; t \in \left(0, \frac{\Delta t}{2}\right) \tag{127}$$

with initial conditions $z(0) = z_0$. Note that unlike the case of unconstrained multibody dynamics (see Eq. (39)), the operation of Eq. (127) depends on the undetermined vector of the Lagrange multipliers λ_0.

Stage 3. Propagate the vector of flexible position variables to the full step $t = \Delta t$ using momenta (recomputed in modal velocities) at $t = \Delta t/2$:

$$\dot{\xi} = U_{\xi_{1/2}}(\lambda_0), \; t \in (0, \Delta t) \tag{128}$$

with initial conditions $\xi(0) = \xi_0$. Note that the dependency of the operation of Eq. (128) on the vector of Lagrange multipliers λ_0 is due to the fact that the vector of absolute velocities is a result of solving a system of linear equations (Eq. (26)) (at $t = \Delta t/2$) where the corresponding momentum vector $p_{1/2}$ depends on λ_0 (see Stage 1).

Stage 4. Propagate the vector of rigid position variables to the full step $t = \Delta t$ freezing the flexible position variables at $t = \Delta t$ and the momentum vector at $t = \Delta t/2$:

$$\dot{z} = V_z\left(z, \xi_1, p_{1/2}(\lambda_0)\right), \; t \in \left(\frac{\Delta t}{2}, \Delta t\right) \tag{129}$$

with initial conditions $z\left(\frac{\Delta t}{2}\right) = z_{1/2}$. Note that this operation unlike its "unconstrained" counterpart (see Eq. (41)) depends on the undetermined vector of Lagrange multipliers λ_0.

It is important to stress that Eq. (126) along with Eqs. (125), and (127)-(129), define a system of nonlinear equations to be solved with respect to the vector of Lagrange multipliers λ_0.

Stage 5. Propagate the momentum vector to the full step $t = \Delta t$ freezing the position states and the vector of Lagrange multipliers at $t = \Delta t$:

$$\dot{p} = G(q_1, p) + \left[g'(q_1)\right]^T \lambda_1, \; t \in \left(\frac{\Delta t}{2}, \Delta t\right) \tag{130}$$

with initial conditions $p\left(\frac{\Delta t}{2}\right) = p_{1/2}$. Note that unlike the case of unconstrained multibody dynamics (see Eq. (42)), now one has to evaluate the vector of Lagrange multipliers λ_1 from the condition of satisfying the bond-length constraints at the velocity level:

$$g'(q_1)D(q_1)M(q_1)^{-1}p_1(\lambda_1) = 0 \tag{131}$$

In other words, one has to solve a L-dimensional system of nonlinear equations for λ_1 defined by Eq. (131) along with Eq. (130).

Note that dealing with the bond-length constraint on the velocity level entails that we actually use the RATTLE (enhanced) version of the SHAKE-type algorithm (see e.g. Ref. 3). However, we will use the name SHAKE as it is common in the literature on integrators. For atomistic MD, according to Ref. 3, the "pure" SHAKE and SHAKE/RATTLE algorithms are both global second order accurate (assuming that the nonlinear equations are solved exactly at each step). The two methods are equivalent at mesh points if initialized appropriately and, moreover, the SHAKE/RATTLE method provides symplectic discretization of the equations of motion ("pure" SHAKE is also symplectic in a slightly weakened sense (see Ref. 3)).

7.3.3 Solution of the Nonlinear Equations

The further concretization of the OMBI/SHAKE algorithm depends on the method of solving a system of nonlinear equations, which arise due to the Lagrange multiplier formalism. Both the system of Eqs. (126), (125), (127)-(129) Stages 1-4 of the OMBI due to the constraints at the position level) and the system of Eqs. (131) and (130) Stage 5 of the OMBI due to the constraints at the velocity level) may be formalized as a system of nonlinear equations

$$f(\lambda) = 0 \tag{132}$$

Since the algorithm for solving this system is to be implemented at each integration step, it should be sufficiently fast in order to make the computational overhead associated with enforcing the constraints much smaller than the reduction in the total CPU time resulting from a longer integration step. This goal can be met thanks to the fact that we need to correct only a small deviation in the bond-length constraints that took place during a single integration step (it is assumed that the constraints were maintained at the previous integration steps). References 3 and 58 offer an excellent analysis of possible options to solve this problem.

First, the authors consider a classical SHAKE iteration (see Ref. 34) based on the recursive coordinate resetting to satisfy the bond-length constraints one by one. Second, they demonstrate that the SHAKE iteration, originally derived in Ref. 34 as a well customized algorithm for a particular problem, is nothing more than an ordinary Gauss-Seidel-Newton (GSN) method to solve a system of L nonlinear equations by decomposing the latter into L scalar problems. Third, using a general formalism of the GSN method, Refs. 3 and 58 offer a modification of the SHAKE algorithm based on Successive Over Relaxation (SOR). Finally, the authors offer a "direct" Newton-Raphson method for solving a system of nonlinear equations in matrix form while taking full advantage of the sparsity in the Jacobian matrix.

In Refs. 3 and 58 the authors consider the Symmetric Newton-Raphson Iteration (SNIP) as an alternative to the conventional Newton-Raphson Iteration (NIP). Motivation for this method comes from the fact that the symmetric approximation of the Jacobian matrix is nearly constant over the coarse of the numerical integration, and hence rarely requires update and refactorization. A general conclusion of the work in Refs. 3 and 58 is that the SHAKE-type iteration with SOR is the most efficient algorithm to maintain bond-length constraints during the integration process. For example, it is up to 3 times faster than the most "exact" NIP iteration or

is about 1.5 faster than SNIP (assuming the same level of tolerance in maintaining constraints in all cases). In other words, the SHAKE-type (or to be more general, GSN-type) scalar decomposition of the system of nonlinear equations turns out to be an efficient technique for local corrections of the bond-length constraints at each integration step.

We will utilize the SHAKE-type iteration with SOR in the OMBI/SHAKE algorithm for constrained multibody dynamics. The SHAKE iteration proceeds as follows (see e.g. Ref. 58). We begin by initializing $\lambda = 0$. Each step of the outer iteration consists of cycling through the L constraints, approximately satisfying each constraint. It is customary to denote by λ_l^k the approximation to λ computed at the k-th step of the outer iteration and the l-th stage of the inner iteration (corresponding to each l-th bond). At the *k*-th step of the outer iteration, the *l*-th constraint is linearized about λ_{l-1}^k, the most recent approximation to λ. The approximation is then corrected in the direction of the gradient in order to satisfy the *l*-th constraint.

In other words, following the more general formalism of the GSN method, one has to solve a scalar nonlinear equation

$$f_l\left(\lambda_1^k \ \ldots \ \lambda_{l-1}^k,\ \lambda_l^k,\ \lambda_{l+1}^{k-1} \ldots \lambda_l^{k-1}\right) = 0 \tag{133}$$

for λ_l^k at each k-th outer iteration and each l-th inner stage. This solution may be formalized in the form

$$\lambda_l^k[i+1] = \lambda_l^{k-1} - \mu \frac{f_l\left(\lambda_l^k[i]\right)}{f_l'\left(\lambda_l^k[i]\right)} \tag{134}$$

where the index *i* denotes internal iterations to solve the scalar nonlinear equation, and μ is the relaxation parameter which can usually hasten convergence. One wishes to find a μ which is optimal for rapid convergence of the iterative process. This is the essence of Successive OverRelaxation (SOR).

In implementing the OMBI/SHAKE algorithm for constrained multibody dynamics we plan to utilize the adaptive relaxation algorithm developed in Ref. 3. It is based on the observation that in the process of MD one has to repeatedly solve nonlinear systems whose form does not change from step to step (we assume that this observation made for atomistic MD will also hold for substructured MD). Thus one can use the behavior of iteration during the early stages of the integration to find a good value of the relaxation parameter μ via iterative improvement. The adaptive relaxation algorithm may be formulated as follows (see Ref. 3):

$\mu_0 = 1$

$\gamma_0 = 0$

$\Delta = \Delta_0$

Initialize:

FOR k = 0,1, ... (time steps)

integrate and perform SHAKE/ SOR with relaxation parameter μ_k

compute average convergence factor γ_k (see text below)

IF $\gamma_k > \gamma_{k-1}$

$$\Delta = -\frac{\Delta}{2}$$

END IF

$\mu_{k+1} = \mu_k + \Delta$

END FOR

Note that in the above algorithm Δ_0 is a parameter (Ref. 3 recommends the use of $\Delta_0 = 0.1$). Also, the convergence factor γ might be taken as the number of iterations required for convergence or as the ratio of iterates $\frac{\Delta\lambda_l^{k+1}}{\Delta\lambda_l^k}$.

In concluding this Section, we would like to mention that substructuring the molecule into a set of bodies significantly reduces the total number of bond-length constraints in the system (compared to an atomistic representation). This will result in a more efficient SHAKE-type algorithm due to a shorter recursive sweep over the constraints (i.e. due to a smaller number of scalar nonlinear equations to solve).

7.3.4 Requirements for the Design of the OMBI/SHAKE Algorithm

The results presented in Section 7.3.1 and Section 7.3.3 provide the principal framework for the OMBI/SHAKE algorithm. In order to have the final algorithm for coding, one has to provide the analytical expressions for Eqs. (126), (125), (127)-(129) needed to numerically integrate the sub-systems of differential-algebraic equations at all five stages of the OMBI integrator. Equations (126), (125), (127)-(129) parametrically depend on the vectors of Lagrange multipliers λ_0 and λ_1. To obtain one has to evaluate all the derivatives involved in the formalism of constrained multibody dynamics. This includes the evaluation of the Jacobian matrix in Eqs. (126), (131) and (130), as well as the derivative of the constraint nonlinearity with respect to the Lagrange multiplier in the SHAKE-type iteration of Eq. (134).

It should be noted that the SHAKE algorithm based on the scalar decomposition of the vector constraint, significantly simplifies the operations mentioned above. Indeed, since the SHAKE algorithm works on each constraint separately, it involves at any step only those blocks in the total state-vector, which correspond, to the two bonded bodies in the system. This differs from the Newton-Raphson iteration, which would require working with all blocks of the state-vector at each step, forming rather complex matrix constructions. As was mentioned in Section 7.3.3, the SHAKE-type iteration is up to 3 times more efficient than the NIP iteration implemented even with sparse-matrix optimization (according to Ref. 3). This conclusion was made for atomistic MD. It is quite realistic to assume that in the case of SMD the difference in the CPU time between the SHAKE and NIP iterations would be much larger due to the more complex matrix operations in the case of multibody dynamics. SHAKE-type scalar decomposition helps to reduce the size of the "matrix" constructs. One of the challenges for Phase II will be to prove in practice that the simplification of the Newton iteration to the SHAKE-type (Gauss-Seidel-Newton) iteration in the case of multibody dynamics still fits within the paradigm of "slight" corrections of the bond-length constraints on each integration step (as it does for atomistic MD).

The final concretization of the OMBI/SHAKE algorithm mainly involves routine procedures like taking derivatives. Taking into account the routine character of the final derivations of the OMBI/SHAKE algorithm we will consider only the principal guidelines for those derivations in this

Section. During Phase II we will document the final OMBI/SHAKE algorithm in separate technical notes and will implement it in FORTRAN for MBO(N)D.

The main principle in evaluating all the necessary derivatives for the OMBI/SHAKE algorithm consists in using a chain rule. This is possible due to the fact that the dependency of the "output" on the "input" is formalized by a sequence of embedded functions and there are analytical expressions for those functions at each level.

Analytical derivation of the Jacobian matrix $g'(q)$ is rather straightforward and involves two-level differentiation of the bond-length nonlinearity defined by Eq. (119). For convenience of notations we will formalize the two-level embedding of nonlinear functions as follows:

$$\begin{aligned} g(q) &= G(x) \\ x &= X(q) \end{aligned} \tag{135}$$

Here, q is the state-vector of the multibody system (see Eq. (116)). The function $G(\cdot)$ represents the dependency of the bond-length constraints on the absolute Cartesian coordinates x of the two atoms making the bond (see Eq. (119)). The function $X(\cdot)$ formalizes the mapping of the state-vector q into the vector x. Note that the mapping operation depends on the type of body each atom belongs to (particle, rigid body, or flexible body). The corresponding nonlinear relations are defined by Eqs. (120)-(122).

Taking all the above into account, the chain rule yields the following result:

$$g'(q) = \frac{\partial G}{\partial x}\frac{\partial X}{\partial q} \tag{136}$$

According to Eq. (119), evaluation of the matrix $\frac{\partial G}{\partial x}$ involves differentiation of the quadratic function. According to Eqs. (120)-(122), evaluation of the matrix $\frac{\partial X}{\partial q}$ also involves differentiation of polynomial vector functions with maximal order of 3. Note that the latter is true since we use Euler parameters for body orientation descriptions. Indeed, the rotational matrix $C(\beta)$ in Eq. (121) is a quadratic function of the corresponding Euler parameters β (which make up one of the blocks in the state vector q). In the case of flexible bodies this quadratic function is "coupled" with the modal amplitudes ξ (which make up another block in the state vector q). This "coupling" yields the polynomial function of the 3rd-order which is still easy to differentiate. It should be mentioned that, while forming matrices and performing matrix multiplication in Eq. (136), one should take advantage of the matrix sparsity (which results from the fact that each bond connects only two bodies).

Analytical evaluation of the scalar derivative $f_i'\left(\lambda_i^k[i]\right)$ in the SHAKE iteration of Eq. (134) also involves a chain differentiation. Consider the corresponding formalism only for the case when the function f represents the bond-length constraint at the position level. In this case f stands for $g(\lambda_0)$ (see Eq. (126) which "couples" stages 1-4 of the OMBI). The second case when f represents the bond-length constraint at the velocity level (see Eq. (131) that arises in g stage 5 of the OMBI) is, in principle, similar to the first case and will be omitted here.

In the considered case, the embedding of nonlinear functions is the following

$$
\begin{aligned}
g(\lambda_0) &= G(x_1) \\
x_1 &= X_1(z_1, \xi_1) \\
z_1 &= Z_1(z_{1/2}, \xi_1, p_{1/2}) \\
\xi_1 &= \Xi_1(p_{1/2}) \\
z_{1/2} &= Z_{1/2}(p_{1/2}) \\
p_{1/2} &= P_{1/2}(\lambda_0)
\end{aligned}
\tag{137}
$$

It should be noted that due to scalar decomposition of the SHAKE iteration one has to work only with a single element of the vectors and λ_0 while evaluating the derivative $g'(\lambda_0)$. The latter significantly simplifies implementation of the chain rule for the sequence (137) since it is sufficient to deal only with those blocks of the related vectors x, z, ξ, and p that correspond to the two bodies making the current bond. Note that in Eq. (137) the indexes 0, ½, and 1 denote the beginning, the middle, and the end of the time step. The nonlinear functions for all six levels in Eq. (137) are described in Section 7.3.1 and Section 7.3.3. The 6th level in Eq. (137) formalizes the propagation of the momentum vector p at the first half step (see Eq. (125), Stage 1 of the OMBI). The 5th level formalizes the propagation of the rigid states z at the first half step (see Eq. (127), Stage 2 of the OMBI). The 4th level formalizes the propagation of the flexible states ξ at the full step (see Eq. (128), Stage 3 of the OMBI). The 3rd level formalizes the propagation of the rigid states z at the second half step (see Eq. (130), Stage 4 of the OMBI). The 2nd level formalizes the transformation of the body's rigid and flexible states into the absolute Cartesian coordinates of the atoms making the bond (see Eqs. (120)-(122)). The 1st level formalizes the bond-length constraint as a function of the absolute Cartesian coordinates of the atoms (see Eq. (119)).

Given the above nonlinearities, one can implement the following chain rule:

$$
\begin{aligned}
g'(\lambda_0) &= \frac{\partial G}{\partial x_1}\frac{\partial X_1}{\partial \lambda_0} \\
\frac{\partial X_1}{\partial \lambda_0} &= \frac{\partial X_1}{\partial z_1}\frac{\partial Z_1}{\partial \lambda_0} + \frac{\partial X_1}{\partial \xi_1}\frac{\partial \Xi_1}{\partial \lambda_0} \\
\frac{\partial Z_1}{\partial \lambda_0} &= \frac{\partial Z_1}{\partial z_{1/2}}\frac{\partial Z_{1/2}}{\partial \lambda_0} + \frac{\partial Z_1}{\partial \xi_1}\frac{\partial \Xi_1}{\partial \lambda_0} + \frac{\partial Z_1}{\partial p_{1/2}}\frac{\partial P_{1/2}}{\partial \lambda_0} \qquad (138) \\
\frac{\partial \Xi_1}{\partial \lambda_0} &= \frac{\partial \Xi_1}{\partial p_{1/2}}\frac{\partial P_{1/2}}{\partial \lambda_0} \qquad (112) \\
\frac{\partial Z_{1/2}}{\partial \lambda_0} &= \frac{\partial Z_{1/2}}{\partial p_{1/2}}\frac{\partial P_{1/2}}{\partial \lambda_0}
\end{aligned}
$$

It should be noted that all nonlinear functions in Eq. (138) are expressed in closed form that is amenable for analytical differentiation. Those functions are rather simple for the 1st and 2nd levels (polynomial functions up to the 3rd order). But even the functions at the 3rd to 6th levels have elegant analytical expressions (see Section 6.1.4, which concretizes the stages of the OMBI). In the practical implementation of Eq. (138) one has to account for scarcity in all matrix operations. The sparsity comes both from the fact that only the coordinates related to two bodies are involved for each bond as well as from the fact that the translational, rotational, and deformational (due to structural flexibility) matrix constructions are uncoupled to some extent in multibody dynamics.

7.4 Algorithms for Initialization of Constrained Dynamics

The important functionality needed for implementation of the constrained multi-body dynamics, is initialization of those dynamics. The latter entails that the bond-length constrains between bodies (both at the position and velocity levels) should be satisfied at the beginning of the MD simulations ($t = 0$).

It is important to stress that from a mathematical point of view the problem of initialization is very similar to the problem of maintaining of the constrains during integration via the OMBI/SHAKE algorithm. We took advantage of this similarity by designing reusable classes in C++. Below these C++ classes are described from an algorithmical point of view.

The VelocityFitter class is designed to interpret both the atomistic system and the substructured system (which is contained in a DynamicSystem object). The purpose of this class is to find the best fit of body velocities that will reproduce as close as possible the atomic velocities, while maintaining any bond-length velocity constraints and maintaining the exact system momenta as in the original atomistic system.

Variable List

N_{Va}^b = Number of atom velocities associated with a particular body, b = $3N_A^b$

N_A^b = Number of atoms in body b

N_X^b = Number of velocities in body b, = 3 translational for particle, 6 (trans + ang) for rigid body

N_A = Total number of atoms in system

N_X = Total number of body velocities in system

N_{VA} = Total number of atom velocities in system = $3N_A$

A^b = Matrix that takes body b velocities and transforms into the bodies specific atom velocities

A = Block diagonal matrix that takes the set of all body velocities and transforms into the set of all atom velocities.

V_A^b = atom velocity vector for all atoms in body b, of dimension $3N_{VA}^b \times 1$

C^b = Transformation matrix for body b that transforms inertial coordinates into body-based coordinates

s_i^b = internal, body-frame relative coordinates of atom i in body b

N_B = Total number of bodies (entities = particles and rigid bodies) in system

X_b = The body velocity vector = $\{v_b \quad \omega_b\}^T$ for rigid bodies

X = The set of all body velocity vectors

7.4.1 Solution of Body Velocities From Atom Velocities

Determining the body velocity vector that best reproduces the inertial atomic velocities of the atoms in a body start with the equations for computing the atom velocities from body velocities. We show the equations for a rigid body only, since the transformation for a particle is one-to-one, i.e., the atom velocity for a particle IS the body velocity. For an atom i in body b, the

velocity of the atom is the sum of the body velocity (inertial frame translational velocity) and the cross-product of the body angular velocity with the atom relative position, projected into the inertial frame,

$$\begin{aligned} V_{Ai}^b &= v_b + \left[C^b\right]^T \omega_b \times s_i^b \\ &= v_b - \left[C^b\right]^T s_i^b \times \omega_b \\ &= v_b - \left[C^b\right]^T \left[\tilde{s}_i^b\right] \omega_b \end{aligned} \tag{139}$$

which can be written in the form of a matrix operator acting on the rigid body velocity vector X_b as,

$$V_{Ai}^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[-C^{bT} \tilde{s}_i^b\right] X_b = A_i^b X_b \tag{140}$$

The matrix A^b transforming all atoms in the body to atomic velocities in the inertial frame is given by:

$A^b = \begin{bmatrix} A_1^b \\ A_2^b \\ \vdots \\ A_{N_A^b}^b \end{bmatrix}$ such that the velocities of all atoms in body b is $V_A^b = \begin{Bmatrix} V_{A1}^b \\ V_{A2}^b \\ \vdots \\ V_{A_N}^b \end{Bmatrix} = A^b X_b$

Then, the entire set of atom velocities can be found from the entire set of body velocities by the matrix transformation A, which is a block diagonal matrix of all A^b body matrices,

$$A = \begin{bmatrix} A^1 & & & 0 \\ & A^2 & & \\ & & \ddots & \\ 0 & & & A^{N_B} \end{bmatrix} \text{ and } V_A = \begin{Bmatrix} V_A^1 \\ V_A^2 \\ \vdots \\ V_A^{N_B} \end{Bmatrix}_{N_{VA}\times 1} = A_{N_{VA}\times N_X} X_{N_X \times 1} \tag{141}$$

First consider the unconstrained case. In this case, we are simply looking at the problem of solving for n unknowns from m equations, where m > n. This is a minimization problem: Solve

$$AX = V_A$$

to minimize the cost function of the error, $e = V_A - AX$,

$$J(X) = \frac{1}{2} e^T e = \frac{1}{2} (V_A - AX)^T (V_A - AX) \tag{142}$$

which can be solved by setting

$$\frac{\partial J}{\partial X} = 0 = -A^T V_A + A^T A X \tag{143}$$

which yields the solution for X using the pseudo inverse of A,

$$X = \left[A^T A\right]^{-1} A^T V_A = A^{\dagger} V_A \tag{144}$$

For large systems, there is no problem of inverting large matrices, for due to the structure of A, being block diagonal by body, the solution for X can be solved body by body for X_b,

$$X_b = \left[A^{b^T} A^b\right]^{-1} A^{b^T} V_A^b \tag{145}$$

Now generalize the results to the constrained case. Here, we wish to solve the problem: Solve for X the following system of equations

$$AX = V_A$$

to minimize the cost function of the error, $e = V_A - AX$,

$$J(X) = \frac{1}{2} e^T e = \frac{1}{2}(V_A - AX)^T (V_A - AX) \tag{146}$$

subject to the constraints,

$\phi(X) = 0$

As with the unconstrained case, we come to the equation:

$$A^T A X = A^T V_A \tag{147}$$

which is very similar in form to the dynamics equation: $M\ddot{q} = F$. We can view the matrix $A^T A$ as a pseudo-mass matrix for the system. To satisfy the constraints, we add Lagrange Multipliers, λ, and use the Jacobian matrix of the Constraints, $D = \frac{\partial \phi}{\partial X} = \left[\frac{\partial \phi_i}{\partial X_j}\right]$ to solve the following system of equations,

$$\begin{aligned} A^T A X \;+\; D^T \lambda &= A^T V_A \\ D X &= 0 \end{aligned}, \tag{148}$$

or in matrix form,

$$\begin{bmatrix} A^T A & D^T \\ D & 0 \end{bmatrix} \begin{Bmatrix} X \\ \lambda \end{Bmatrix} = \begin{Bmatrix} A^T V_A \\ 0 \end{Bmatrix} \tag{149}$$

In actuality, there are two sets of constraints, one for bond-length velocities and one for system momenta, and the above problem can be further subdivided into ϕ_V for bond-length constraints and ϕ_Z for momenta constraints. Similarly, we can subdivide the Lagrange multipliers into λ_V and λ_Z, which yields

$$\begin{bmatrix} A^T A & D_V^T & D_Z^T \\ D_V & 0 & 0 \\ D_Z & 0 & 0 \end{bmatrix} \begin{Bmatrix} X \\ \lambda_V \\ \lambda_Z \end{Bmatrix} = \begin{Bmatrix} A^T V_A \\ 0 \\ 0 \end{Bmatrix} \tag{150}$$

We will discuss the solution method for the Lagrange multipliers later. For now, let us assume that we have values for the bond-length and momenta Lagrange multipliers. Then, we are simply solving a modified least-squares equation for X,

$$\begin{aligned} A^T A X &= A^T V_A - D_V^T \lambda_V - D_Z^T \lambda_Z \\ X &= \left[A^T A\right]^{-1} \left\{A^T V_A - D_V^T \lambda_V - D_Z^T \lambda_Z\right\} \end{aligned} \tag{151}$$

Again, the solution of large systems is made simpler by the fact that the above problem can be solved for body by body (once Lagrange Multipliers have been solved), because we can look at the body-by-body contributions of the products of the Jacobians and Lagrange multipliers. This will become evident when we look more closely at computing the Jacobians and their data structures.

7.4.2 Bond-Length Velocity Constraints

The bond-length constraint between an atom on body A and an atom on body B is formed by the statement that the distance computed by the sum of the squares of the differences in atom positions minus the bond-length distance squared must be zero,

$$\begin{aligned} &\phi = \Delta x^2 + \Delta y^2 + \Delta z^2 - d^2 = 0 \\ &\Delta x = x_A - x_B; \Delta y = y_A - y_B; \Delta z = z_A - z_B \end{aligned} \tag{152}$$

The velocity constraint is the time derivative of the above equation,

$$\begin{aligned} &\dot{\phi} = 2\Delta x \Delta \dot{x} + 2\Delta y \Delta \dot{y} + 2\Delta z \Delta \dot{z} = 0 \\ &\Delta \dot{x} = \dot{x}_A - \dot{x}_B; \Delta \dot{y} = \dot{y}_A - \dot{y}_B; \Delta \dot{z} = \dot{z}_A - \dot{z}_B \end{aligned} \tag{153}$$

The equation development of this Section is somewhat figurative in describing the constraints and constraint Jacobian. For Bond-length constraints, the constraints as functions of the body-velocities are actual the time-derivative of the bond-length constraints (which are functions of the body generalized coordinates). If we denote the vector of body generalized coordinates as q and the vector of body generalized velocities as $\dot{q}$, then the variables we are solving for are actually $X = \dot{q}$. Now, the partial derivate of the constraint vector with respect to the generalized coordinates for particles is a relatively simple thing (since the coordinates of the particle are the coordinates of the atom in question). We will look at the case for rigid bodies, which have Quaternions to represent the orientation, yielding 7 generalized coordinates per body. However, the generalized velocities using body-frame angular velocities for orientation rates, and thus there are only 6 generalized velocities for rigid bodies. Therefore, in order to compute the constraint Jacobian for rigid bodies, we use the relationship,

$$\begin{aligned} &\dot{\phi} = \frac{\partial}{\partial t}\phi(q) = \frac{\partial \phi}{\partial q}\frac{\partial q}{\partial t} = \left[\frac{\partial \phi}{\partial q}\right]\dot{q} \\ &\frac{\partial \dot{\phi}}{\partial \dot{q}} = \frac{\partial}{\partial \dot{q}}\left(\left[\frac{\partial \phi}{\partial q}\right]\dot{q}\right) = \left[\frac{\partial \phi}{\partial q}\right] \end{aligned} \tag{154}$$

Using the above equation, if we let $\phi_v = \dot{\phi}$, then $\partial\phi/\partial q = \partial\dot{\phi}/\partial\dot{q} = \partial\phi_v/\partial X$.

If there are N_{Bond} bond-length constraints, then ϕ_v will be a vector of N_{Bond} equations, each equation connecting two entities. Now, the Jacobian for the entire system, $D_v = \partial\phi_v/\partial X$, can be analyzed by looking at, for each constraint, the two bodies the constraint connects. All other body blocks will be zero (partials w.r.t. body variables that are not in the equation are zero). Each body block of the constraint Jacobian will be either 1×3 or 1×6 depending on whether the body is a particle or rigid body, respectively. We can look at an example structure of a constraint Jacobian,

$$
D_v = \begin{array}{c|cccccc}
 & B_1 & B_2 & B_3 & B_4 & \cdots & B_{N_B} \\
\phi_1 & \left[\frac{\partial \phi_1}{\partial X_{B1}}\right]_{1\times N_X^1} & \left[\frac{\partial \phi_1}{\partial X_{B2}}\right]_{1\times N_X^2} & & & & \\
\phi_2 & & \left[\frac{\partial \phi_2}{\partial X_{B2}}\right]_{1\times N_X^2} & \left[\frac{\partial \phi_2}{\partial X_{B3}}\right]_{1\times N_X^3} & & & \\
\phi_3 & & & \left[\frac{\partial \phi_3}{\partial X_{B3}}\right]_{1\times N_X^3} & \left[\frac{\partial \phi_3}{\partial X_{B4}}\right]_{1\times N_X^4} & & \\
\vdots & & & & & & \\
\phi_{Nc} & & & \left[\frac{\partial \phi_{Nc}}{\partial X_{B3}}\right]_{1\times N_X^3} & & & \left[\frac{\partial \phi_{Nc}}{\partial X_{BNB}}\right]_{1\times N_X^{NB}}
\end{array} \tag{155}
$$

The structure of the Jacobian in equation (155) provides insight into how to handle the sparsity of the Jacobian for large systems. We will discuss this further in the section on methods and attributes of the VelocityFitter class.

7.4.3 Momentum Constraint

The purpose of the momentum constraints is to ensure that the resulting fit for body velocities produces the same system momenta as computed from the atomistic system. First, we will discuss the equations for computing system Linear and Angular Momenta from both atom-based and body-based systems (as shown in Figure 9).

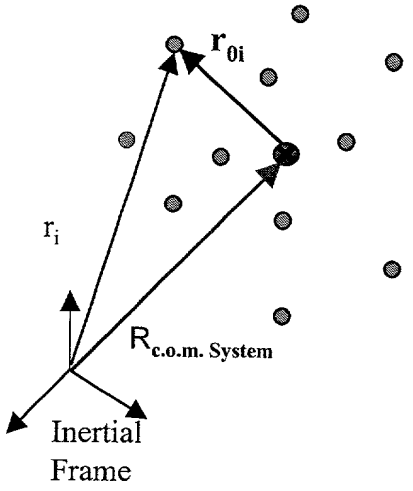


Figure 9. Geometrical Interpretation of Vectors used in Momentum Constraint

For the atom based system, we can find the instantaneous system center of mass from the coordinates and masses of the atoms,

$$r_{Systemc.o\,m} = \frac{\sum r_i m_i}{\sum m_i} \tag{156}$$

and we can calculate the vector from the system center of mass to the i-th atom by

$$r_{Sys\ c.o.m.} + r_{0i} = r_i$$
$$r_{0i} = r_i - r_{Sys\ c\ o.m.} \tag{157}$$

For the following discussion, we define G_0, H_0 as the System Linear and Angular momenta vectors, and $Z_0 = \{G_0^T \ H_0^T\}^T$. We will denote the quanties computed from the atomistic system by the subscript α and the quantities computed from the body-based system by the subscript β.

For the atomistic system, the Linear momentum vector is found by the equation,

$$(G_0)_\alpha = \sum_{\text{allatoms}} m_i v_i \tag{158}$$

and the Angular momentum vector is found from summing the contributions to angular momentum from all atoms in the system about the system center of mass,

$$\begin{aligned}(H_0)_\alpha &= \sum_{\text{all atoms}} m_i\, r_{0i} \times v_i \\ &= \sum_{\text{all atoms}} m_i \left[\tilde{r}_{0i}\right] v_i\end{aligned} \tag{159}$$

System Momentum using Atomic Velocities computed from Body Velocities

In this case, we attempt to make the System Momentum calculations as close as possible to the original atomic-velocity calculations. We compute the atomic velocities from the bodies' velocities. For each body, we use Eq. (140) to map body velocities into its constituent atom velocities, $A_b X_b = V_A^b$. Then, to compute the System Linear momentum, we can use the matrix formulation,

$$G_{0\beta} = \sum_{Bodies} \underbrace{\left[\left[m_1 1_{3\times3}\right] \ \left[m_2 1_{3\times3}\right] \ \cdots \ \left[m_{N_c^b} 1_{3\times3}\right]\right]}_{3\times 3N_c^b} \underbrace{A_b}_{3N_c^b \times N_X^b} \underbrace{X_b}_{N_X^b \times 1} \tag{160}$$

$$H_{0\beta} = \sum_{Bodies} \underbrace{\left[\left[m_1 \tilde{r}_{0_1}\right] \ \left[m_2 \tilde{r}_{0_2}\right] \ \cdots \ \left[m_{N_c^b} \tilde{r}_{0_{Nc}}\right]\right]}_{3\times 3N_c^b} \underbrace{A_b}_{3N_c^b \times N_X^b} \underbrace{X_b}_{N_X^b \times 1} \tag{161}$$

such that the Body-velocity computed System momentum vector can be represented as

$$Z_{0\beta} = \sum_{Bodies} \underbrace{\begin{bmatrix} \left[m_1 1_{3\times3}\right] & \left[m_2 1_{3\times3}\right] & \cdots & \left[m_{N_c^b} 1_{3\times3}\right] \\ \left[m_1 \tilde{r}_{0_1}\right] & \left[m_2 \tilde{r}_{0_2}\right] & \cdots & \left[m_{N_c^b} \tilde{r}_{0_{Nc}}\right] \end{bmatrix}}_{6\times 3N_c^b} \underbrace{A_b}_{3N_c^b \times N_X^b} \underbrace{X_b}_{N_X^b \times 1} \tag{162}$$

Then, the body blocks of the Momentum Constraint Jacobian are computed from the $6\times 3N_c^b$ matrix of Eq. (162). For particles, the mapping matrix A_b is simply the identity, and the Jacobian body block is found from

$$D_z^b = \begin{bmatrix} \left[m_1 1_{3\times3}\right] \\ \left[m_1 \tilde{r}_{0_1}\right] \end{bmatrix} \tag{163}$$

For rigid bodies, the Jacobian body block for Momentum constraints is computed as

$$D_z^b = \underbrace{\begin{bmatrix} \left[m_1^b 1_{3\times3}\right] & \left[m_2^b 1_{3\times3}\right] & \cdots & \left[m_{N_c^b}^b 1_{3\times3}\right] \\ \left[m_1^b \tilde{r}_{0_1}\right] & \left[m_2^b \tilde{r}_{0_2}\right] & \cdots & \left[m_{N_c^b}^b \tilde{r}_{0_{Nc}}\right] \end{bmatrix}}_{6\times3N_c^b} \underbrace{A_b}_{3N_c^b\times N_X^b} \tag{164}$$

Using this method to compute system Momentum from the Body-based velocities has the most success at reproducing the closest approximations to the original atomic-based system. We provide the equations for computing the System momentum strictly from body coordinates, velocities and internal body momenta simply for traceability.

System Momentum using Body-Variables

For the body-based system, the linear momentum is found similarly to the atomistic system,

$$(G_0)_\beta = \sum_{\text{all Bodies}} m_b v_b \tag{165}$$

and the angular momentum is found similarly to the atomistic system, but with any rigid-body's internal angular momentum included as well,

$$\begin{aligned} (H_0)_\beta &= \sum_{\text{all entities}} \left(m_b\left[\tilde{r}_{0b}\right]v_b + \left[I_b\right]\omega_b\right) \\ &= \sum_{\text{all entities}} \left[m_b\left[\tilde{r}_{0b}\right]v_b \qquad \left[I_b\right]\omega_b\right]\{X_b\} \end{aligned} \tag{166}$$

Note that in the above equation, if the entity is a particle, then the entity itself has no angular momentum (no rotational inertia).

The momentum constraint is formulated as

$$\phi_Z = Z_\beta - Z_\alpha \tag{167}$$

where ϕ_Z is a 6×1 vector of equations that are all functions of all entities in the system. While the number of constraints is always fixed at 6 (opposed to the system-dependent case of bond-length constraints), the Constraint Jacobian will not be sparse. This is because every body's coordinates are involved in the constraint equations.

From the above equations, we can see that computing the body-blocks of the Momentum constraint Jacobian is rather straightforward. For particles, we have

$$(D_Z)_b = \begin{bmatrix} m_b\left[1_{3\times3}\right] \\ m_b\left[\tilde{r}_{0b}\right] \end{bmatrix} \tag{168}$$

and for rigid bodies we have

$$(D_Z)_b = \begin{bmatrix} m_b\left[1_{3\times3}\right] & 0 \\ m_b\left[\tilde{r}_{0b}\right] & \left[diag(I_1, I_2, I_3)\right] \end{bmatrix} \tag{169}$$

7.4.4 Iterative Solution of Lagrange Multipliers

To avoid the inversion of possible very large matrices in a "brute-force" solution method in solving for the Lagrange multipliers, we apply an iterative solution similar to the SHAKE algorithm. Here, we start with λ_v and λ_Z equal to zero vectors,

$$\lambda_v^{(0)}=0;\quad \lambda_z^{(0)}=0 \tag{170}$$

We then solve equation (151) for the body velocities, X_b, body-by-body, applying the body-blocks of the constraint Jacobians to incorporate contributions of Lagrange Multipliers,

$$X_b^{(iter)}\left(\lambda_V^{(iter)},\lambda_Z^{(iter)}\right)=\left[A^{bT}A^b\right]^{-1}\left\{A^{bT}V_A^b-D_V^{bT}\lambda_V^{(iter)}-D_Z^{bT}\lambda_Z^{(iter)}\right\} \tag{171}$$

After the body velocities have been solved from Eq. (171), the constraint violations are computed (i.e., we use Eqs. (153) and (167) to compute ϕ_V and ϕ_Z).

$$\begin{aligned}\phi_v&=\phi_v\left(X^{(iter)}\right)\\ \phi_Z&=\phi_Z\left(X^{(iter)}\right)\end{aligned} \tag{172}$$

We then update the Lagrange Multipliers using a gradient method. For the velocity constraints, we evaluate the increment in Lagrange Multiplier constraint-by-constraint, where c = 1,...,Nc,

$$\Delta\lambda_{V_c}=\frac{\phi_{V_c}}{D_{V_c}^{(B1)}\left[A^{B1^T}A^{B1}\right]^{-1}D_{V_c}^{(B1)^T}+D_{V_c}^{(B2)}\left[A^{B2^T}A^{B2}\right]^{-1}D_{V_c}^{(B2)^T}} \tag{173}$$

Note that the denominator in Eq. (173) is a computationally simplified expansion of the full matrix expression for the denominator, $D_{V_c}\left[A^TA\right]^{-1}D_{V_c}^T$. We use the fact that the product A^TA is block diagonal by body, the inverse can be computed block by block and that the constraint Jacobian row for the c-th constraint is only non-zero for the two bodies, B1 and B2, that the constraint connects.

For the Momentum constraints, it is easier to compute the increment in Lagrange Multipliers all at once, for there are always only 6 constraints, and $\Delta\lambda_Z$ will be a 6 x 1 vector,

$$\Delta\lambda_Z=\underbrace{\left[\underbrace{D_Z}_{6\times N_X}\underbrace{\left[A^TA\right]^{-1}}_{N_X\times N_X}\underbrace{D_Z}_{N_X\times 6}\right]^{-1}}_{6\times 6}\underbrace{\phi_Z}_{6\times 1} \tag{174}$$

Again, the block diagonal structure of the matrix A of Eq. (141) allows us to simplify the computations of the matrix product $D_Z\left[A^TA\right]^{-1}D_Z^T$ and its inverse, so that at most we are only inverting 6 x 6 matrices.

Once the Lagrange multiplier increments are computed, we can update the Lagrange multipliers using the formula,

$$\begin{aligned}\lambda_V^{(iter+1)}&=\lambda_V^{(iter)}+\alpha^{(iter)}\Gamma_V\Delta\lambda_V\\ \lambda_Z^{(iter+1)}&=\lambda_Z^{(iter)}+\alpha^{(iter)}\Gamma_Z\Delta\lambda_Z\end{aligned} \tag{175}$$

where α is a relaxation parameter that gradually introduces the increment, starting at 0 when iter=0 and increasing to 1, and Γ is an adaptive gain that is either halved or doubled depending on the rate of convergence of the constraints to zero. A good value for the Γ's to begin with is 0.5. During the iterative process, if $\left\|\phi_V^{(iter+1)}\right\|>\left\|\phi_V^{(iter)}\right\|$ then we multiply Γ_V by 0.5. If in the process

of iteration, $\left\|\phi_V^{(iter)}\right\| - \left\|\phi_V^{(iter+1)}\right\| < tolerance$, we increase Γ_V by a factor of 2. The same procedure is performed for Γ_Z. This method has produced the most consistent convergence rates so far.

7.5 Relation of OMBI and OMBI/SHAKE to Known Symplectic Integrators

The Optimized Multibody Integrator (OMBI) is derived by using the RESPA-type framework of Ref. 45, which in its turn takes its roots in the general and well-known symmetrization methods for integrating the systems of differential equations. Armed with this general methodology, one has to apply it to the particular structure of differential equations. The OMBI is a result of this application, which utilizes the benefits of the Euler parameter formulation for multibody dynamics and is the first (to the extent of our knowledge) integrator, which is directly tailored to the Euler parameter formulation.

At the same time, in recent years there have been a number of publications on symplectic integrators for rigid body dynamics (Ref. 16, 17, and 49). It is interesting to compare the OMBI to these integrators. Note that in the following we will not consider the issue of constraints between bodies since the OMBI can incorporate constraints in a way similar to the symplectic integrators of Refs. 16, 17, and 49 (SHAKE-type solution) or utilizing the MBO(N)D's O(N)-algorithm. First, it should be mentioned that the OMBI is a more general integrator since it is implemented for the case of flexible bodies. As a result of this, we also use a more general (non-diagonal) mass matrix which formalizes the couplings between rotational, translational, and deformational motions. One of the most important differences between the OMBI and the symplectic rigid-body integrators of Refs. 16, 17, and 49 consists in the fact that the OMBI, as was mentioned above, effectively utilizes the benefits of the Euler parameter formulation. In Ref. 49 the rotational motion is described by the 3×3 rotational matrix Q (which is a state variable with the corresponding conjugate momenta matrix P). In this case, as shown in Ref. 49, the Hamiltonian is separatable, which automatically leads to a symplectic Verlet-type integrator. Although this integrator is simple in notation, its inconvenience consists in the fact that one has to satisfy an additional constraint $Q^T Q - I = 0$. The authors of Ref. 49 showed that by using the reduced vector of angular momenta p_ω (in body frame) instead of the conjugate momenta matrix P, it is possible to obtain equivalent differential equations for p_ω and Q (instead of equations for P and Q) and avoid the need for the constraint $Q^T Q - I = 0$. But, one still needs to propagate the whole matrix Q (note that the propagator is based on additional Trotter decompositions and evaluation of the corresponding matrix exponentials at each stage of the propagator). The OMBI, on the other hand, simply reduces the propagator for the redundant matrix Q to the equivalent propagator for the Euler parameters e. This propagator does not involve additional Trotter decompositions (the Euler parameters are propagated in the vector form) and, besides, we found a very simple analytical form for the corresponding matrix exponential.

It is interesting to note that Refs. 16 and 17 directly use particulation of rigid bodies in order to derive symplectic integrators for rigid-body dynamics. In this sense, the approach of Refs. 16 and 17 is even more natural than that of Ref. 49. Indeed, in this case one directly uses Cartesian coordinates (and the corresponding velocities) of the particles, which represent the body. As shown in Ref. 16 and 17 the rigid body can be represented by four particles and all interaction forces between bodies may be applied to those 4 points. In the particulation approach no matrix Q is involved and, correspondingly, instead of the constraint $Q^T Q - I = 0$ one uses the natural distance constraints, which are introduced due to the definition of a rigid body. In this case, the Hamiltonian is separatable which leads to a Verlet-type integrator for constrained dynamics (the RATTLE algorithm).

Both the particulation approach of Refs. 16 and 17 and the "Q-matrix" approach of Ref. 49, although as not general as the OMBI, provide excellent theoretical insights into why the time-symmetric integrators have high performance. Namely, they prove that in certain ("redundant") coordinates the Hamiltonian is separatable and the Verlet-type integrator based on the Trotter decomposition (symmetric splitting of the Liouville operator) is symplectic. At the same time, Ref. 49 uses the reduction of variables $P \rightarrow p_w$ to simplify the final equations. Following the arguments for Refs. 49, 16, and 17, it seems likely that the OMBI can be shown to be symplectic for both rigid and flexible bodies. One of the suggestions in Phase II Proposal was to prove this fact from a strict theoretical point of view (following Refs. 16 and 17). But, we concentrated our effort primarily on the implementation of the OMBI and OMBI/SHAKE in the OO C++ MBO(N)D-II. Correspondingly, we were satisfied by the excellent algorithm performance in the real-world large-scale MD.

8 Task 4: Software Implementation of OMBI to Create Powerful New Version of MBO(N)D

8.1 Why Object-Oriented C++?

The best way to implement the advanced multi-body dynamics in a commercial software (upgraded MBO(N)D code) is to use OO C++ programming approach. This was especially driven by the specifics of the multi-body system, which consists of many various dynamic entities (particles, rigid bodies, flexible bodies). In addition, new types of dynamic entities such as continuum or stochastic-boundary may be added (although, they are out of the scope of the AFOSR project). The interactions and constraints between objects can be also thought as objects (note that interactions are performed via a forcefield while the constraints via geometrical relations). In general, the overall picture of the system including integrators looks rather diverse in terms of objects involved. How to manage all those objects in the best way? The natural answer is to define the C++ classes in such a way that the instances of those classes would formalize natural objects in the system.

Given this general framework one can assemble the final system from C++ classes much like Figure 10 is assembled to illustrate the major functionality of the system. First, one can formalize dynamics of a molecular system (or, potentially, any other system, e.g. spacecraft) as a motion of particles, rigid bodies and flexible bodies. Second, different interactions between dynamic entities according to a standard molecular dynamics forcefield can be modeled. Third, to remove high-frequency bond interactions, one can impose hard constraints between bodies formalized by geometrical functions (e.g. distance between two constituent atoms). Fourth, one can account for a real frequency distribution in the system by assigning different time steps to dynamics, interactions (for each type separately) and constraint samplings.

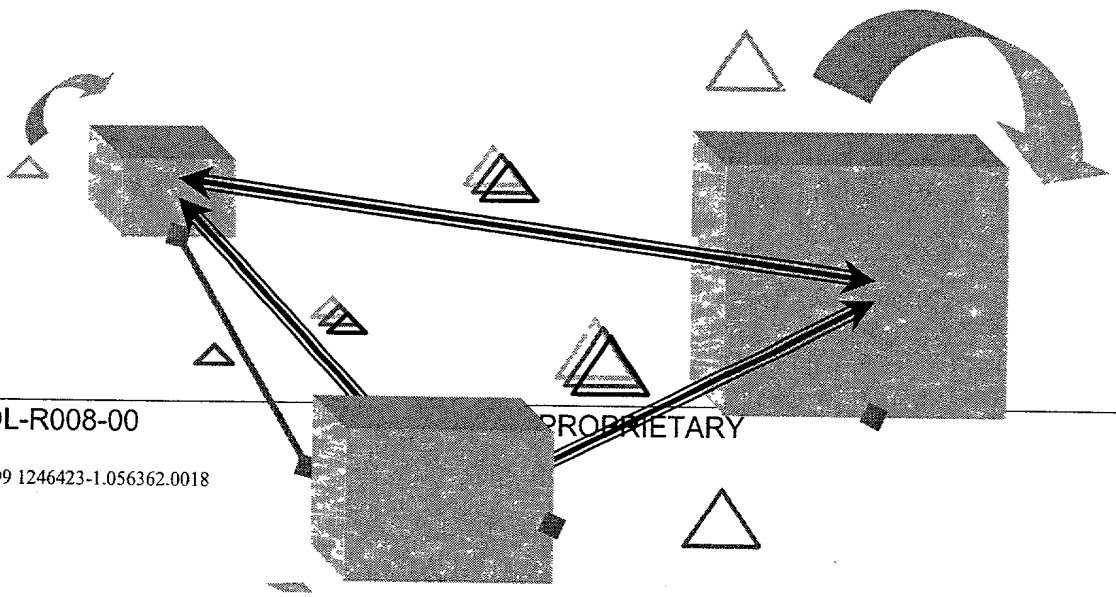


Figure 10. Modeling Multi-Body System

The main advantage of OOP C++ is in the use of encapsulation and data hiding which make it possible to keep the data (attributes) and methods to work on this data (operations) into a single construction, a class. This removes the main disadvantage of structural programming when one has to send the data to a separate function to process it. It took a quarter of century in programming practice to figure out that the separation of data and methods is the main culprit in writing poorly-maintainable, non-reusable, etc. codes. In nature, data and methods are not separable (case in point, a brain cell). The OOP C++ capability of encapsulation and data hiding extended and enhanced by inheritance and polymorphism made it possible to be closer to natural arrangement of things. As a result, this dramatically improved the code's maintainability, reusability, etc.

Much like OOP is superior to structural programming due to coupling data and methods, the OMBI is an optimal integrator due to coupling dynamics and integration. In other words, the OMBI is the model-based integrator directly tailored for multi-body dynamics by incorporating built-in analytical operations on dynamical equations. This is in contrast to the model-independent and general integrators (e.g. Lobatto), which use only accelerations as a model

output. C++ makes it possible to design a class, which encapsulates both parameters related to a dynamic entity (e.g. rigid body) and the methods (function) how those parameters change during integration.

It should be emphasized that only model-based integrators, which are tailored for multi-body dynamics, can provide a complete set of features illustrated in Figure 10 and ensure high performance in terms of speed, accuracy and conservation laws (conservation of total energy and momenta in the system). Model-independent integrators (such as Runge-Kutta integrators) are not sufficient for the task, e.g., they have a drift in energy and momenta, they cannot properly embrace the MTS formalism (which is the key for speed), etc. As one more example, it should be noted that by introducing OMBI as a model-based integrator Moldyn, Inc. made it possible to automatically satisfy an unity-norm constraint on 4-dimensional vector of quaternions for each body.

In short, the advanced developments in multi-body dynamics and the most advanced programming framework (OO C++) are naturally combined in this AFOSR project.

8.2 Overview of Object-Oriented C++ Design

The purpose of this section is to describe the design of the Stage I Object-Oriented (OO) C++ Prototype for the development of the Optimal Multi-Body Integrator (OMBI) software.

The implementation of object-oriented technology, in the development of the OMBI software, is being done incrementally, and in Stage I we only seek to perform Lobatto integration for particles and rigid bodies using constant energy simulations, with no constraint conditions. We have decided to use Absolute Euler angles and velocity variables in the Stage I design, and since there are no constraints, the Newton-Euler equations will be used to propagate the equations of motion. The methods for body numbering and substructuring will remain in FORTRAN, as we do not seek to change the system initialization method in this stage. As input the OO Prototype will require the geometrical and inertial properties of the bodies, as well as the MBO(N)D initial conditions. This data will be obtained using the existing FORTRAN code.

MBO(N)D requires a Molecular Modeling Host (MM Host), presently that host is CHARMm [18]. In order to maintain consistency with the overall design, the term MM Host will be used when referring to CHARMm, since CHARMm is the Molecular Modeling Host. In stage I, some of the functionality of MBO(N)D is left in FORTRAN, this is mainly in the initialization of the system. Since this occurs outside of the OO Prototype, both the MM Host and the FORTRAN portion of the MBO(N)D are viewed as a single external unit.

8.3 Decomposition Description

This section gives a brief description of the overall design of the OO Prototype we accomplished in Year 2. This description also includes the entities that compose the OO Prototype. The OO Prototype operates in a mixed-language environment, which imposes certain constraints on the design of the OO Prototype system. FORTRAN cannot directly access objects in C++, therefore FORTRAN functions that need to access Objects in C++ must call C++ functions, which can then access the C++ objects.

The C++ OO Prototype is comprised of four subsystems. These subsystems are the C++ functions called from FORTRAN, to access the OO Prototype, the Host Interface, the Physical System, and the Simulation Package. The figure below shows the subsystems, and how they interact.

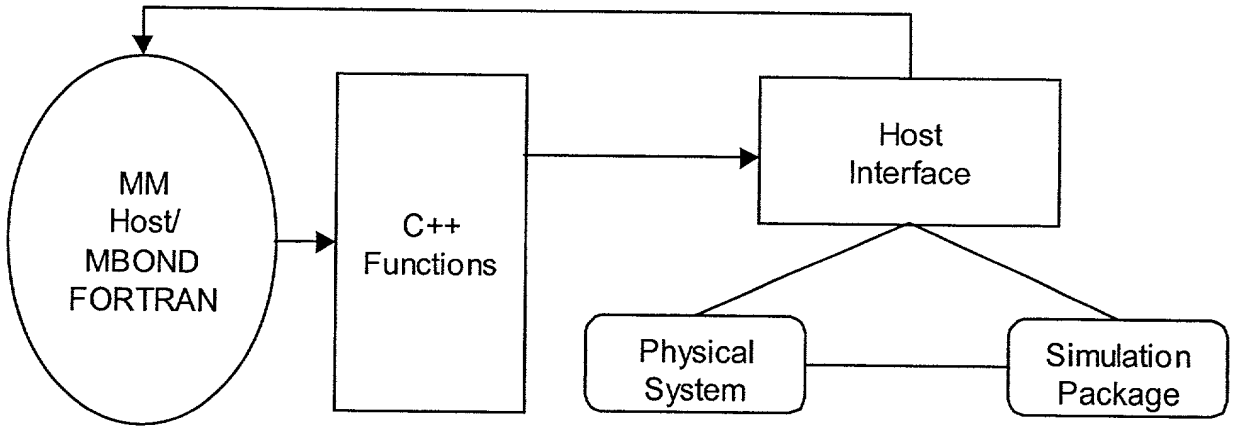


Figure 11. OO Subsystem Interaction

The MM Host/MBO(N)D FORTRAN represents a client of the system. The C++ Functions allow the FORTRAN portion of the system to interact with the Object-Oriented C++ portion of the system. The C++ functions are placed in a single file. The Host Interface is the subsystem that allows communication between the client of the system and the OO Prototype. The Physical System represents the molecular system that is being simulated. It contains information about the geometrical properties and the dynamic variables of the molecular system. The Simulation Package is the subsystem that is responsible for the simulation. It handles such task as the integration loop and periodic events that occur during the simulation.

8.4 UML Design of the OO C++ MBO(N)D-II Code

8.4.1 Overall Class Diagram

A general scheme of the OO subsystem interaction shown in Figure 11, was used a guide in creating the overall class diagram in the Rational Rose UML (see Figure 12).

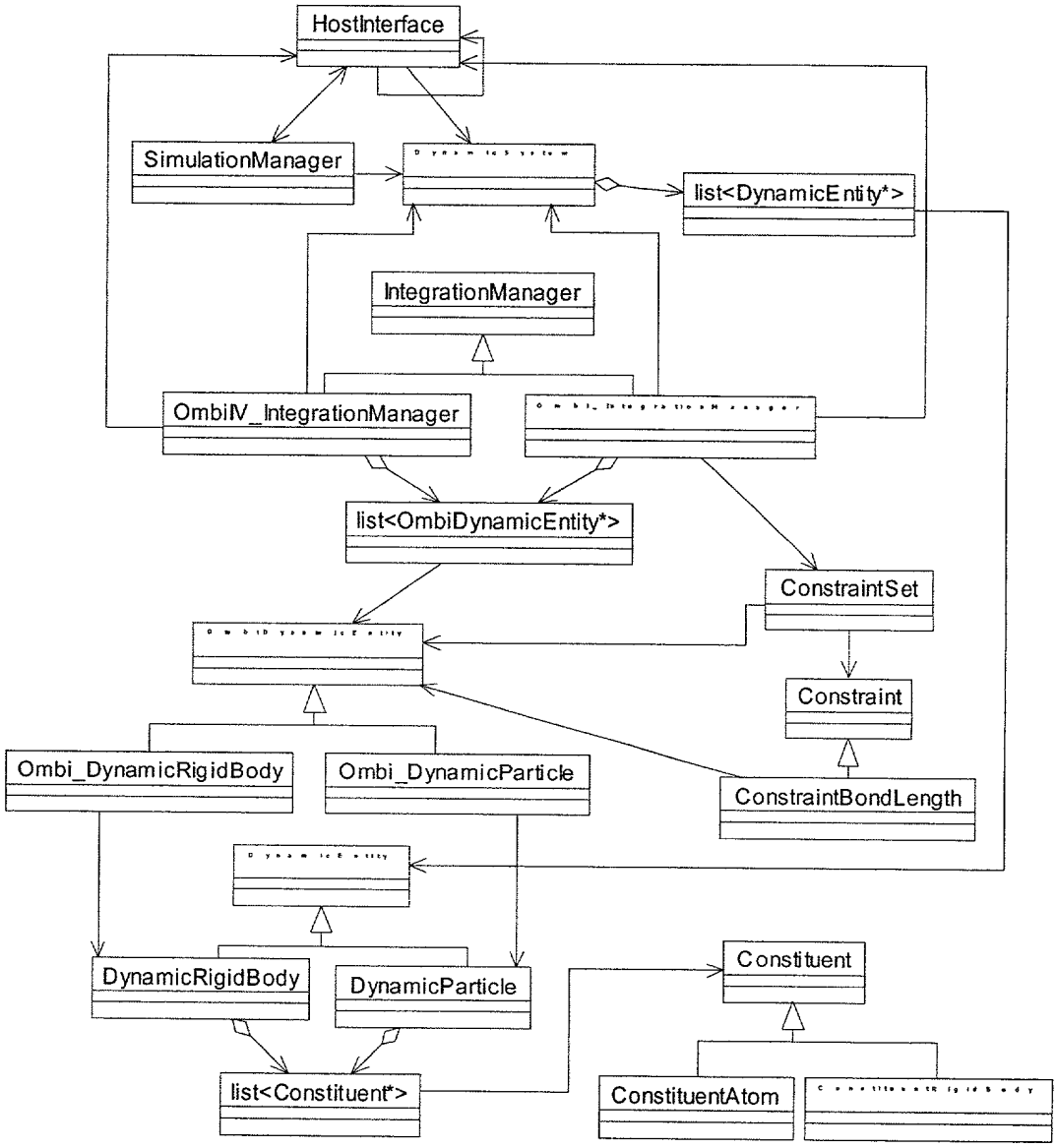


Figure 12. Class Diagram for C++ Classes

The C++ classes are categorized as follows:

"Building Block" Classes

Constituent, ConstituentAtom and ConstituentRigidBody

DynamicEntity, DynamicParticle, DynamicRigidBody

"System" Classes

DynamicSystem, SimulationManager, HostInterface

Integration Classes

IntegrationManager, Ombi_IntegrationManager

OmbiDynamicEntity, Ombi_DynamicParticle, Ombi_DynamicRigidBody

OmbiIV_IntegrationManager

Constraints

Constraint, ConstraintBondLength, ConstraintSet

The C++ classes are accompanied by "wrapper" C-functions that establish an interface to the FORTRAN code.

8.4.2 Sequence Diagram for Initialization / Creating a Multibody System

As it stands now, the multibody system is created and initialized form information passed through to the C++ code from the Fortran-MBOND initialization part (specifically, mbdyna_1(...) passes the system info). Information on atom groupings (into bodies), list of all atoms, atom masses and coordinates, initial body positions, rotations and velocities are all in 1D arrays in Fortran. The memory addresses where this info resides is passed through to the C++ code using "wrapper" functions which the FORTRAN subroutine mbdyna_1 may access. This information is rearranged into more intuitive multidimensional arrays to be passed to C++ functions. The sequence diagram for initializing a system from mbdyna_1(...) is given in Figure 13.

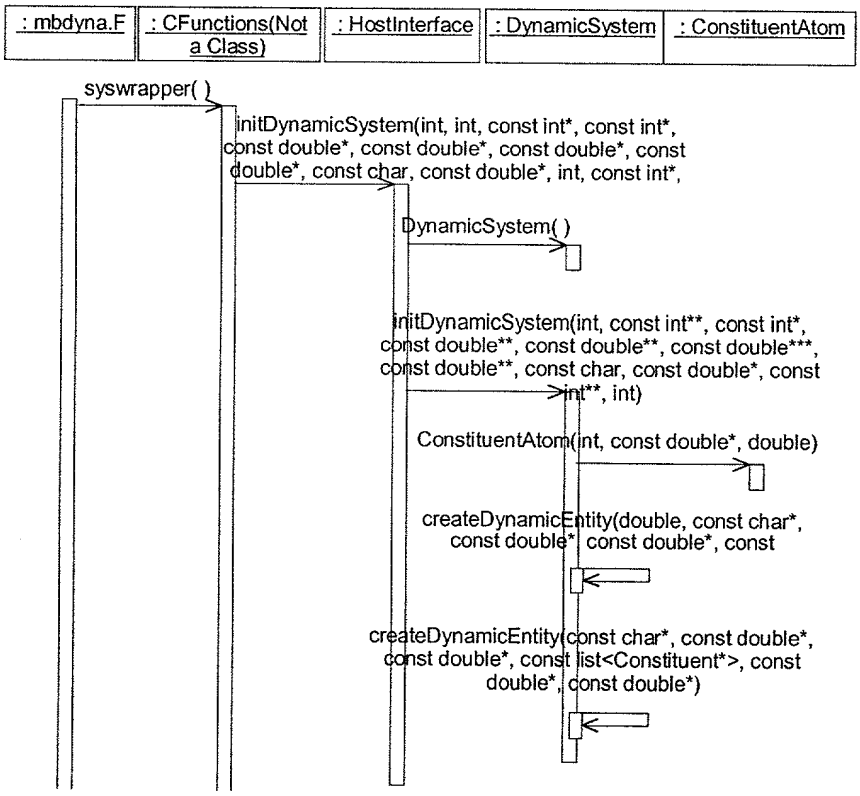


Figure 13. Initialization of System from FORTRAN MBOND function, mbdyna_1.

8.4.3 Sequence Diagram for Integration of Multibody System

The requirements for the design of the OMBI integration process are described in a detail in Section 6.2. Figure 14, Figure 15, and Figure 16 formalize this design in a form of the UML

sequence diagrams. Correspondingly, Figure 14 provides a sequence diagram for starting the integration process (after it was initialized according the sequence diagrams described in Sections 8.4.2 and 8.4.3). Figure 15 provides a sequence diagram for the integration process with a specified number of steps. Figure 16 provides a sequence diagram to describe operations for a single integration process.

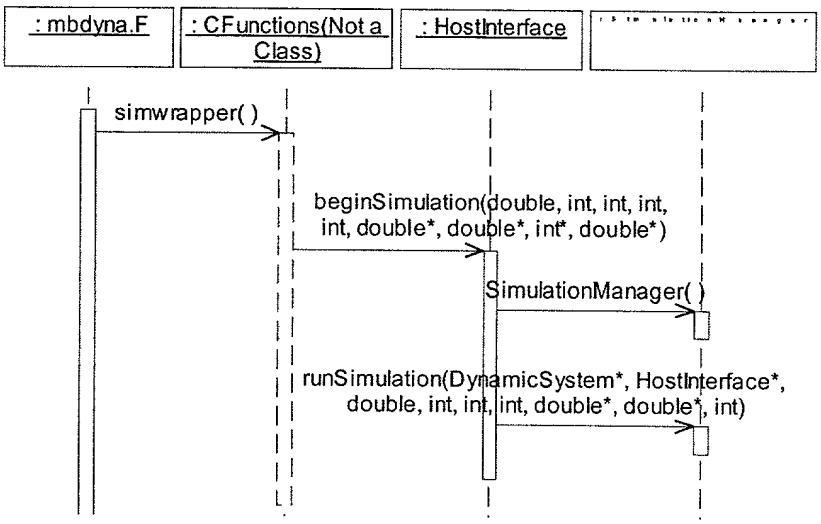


Figure 14. Sequence Diagram for Starting Simulation from Fortran mbdyna_1(...)

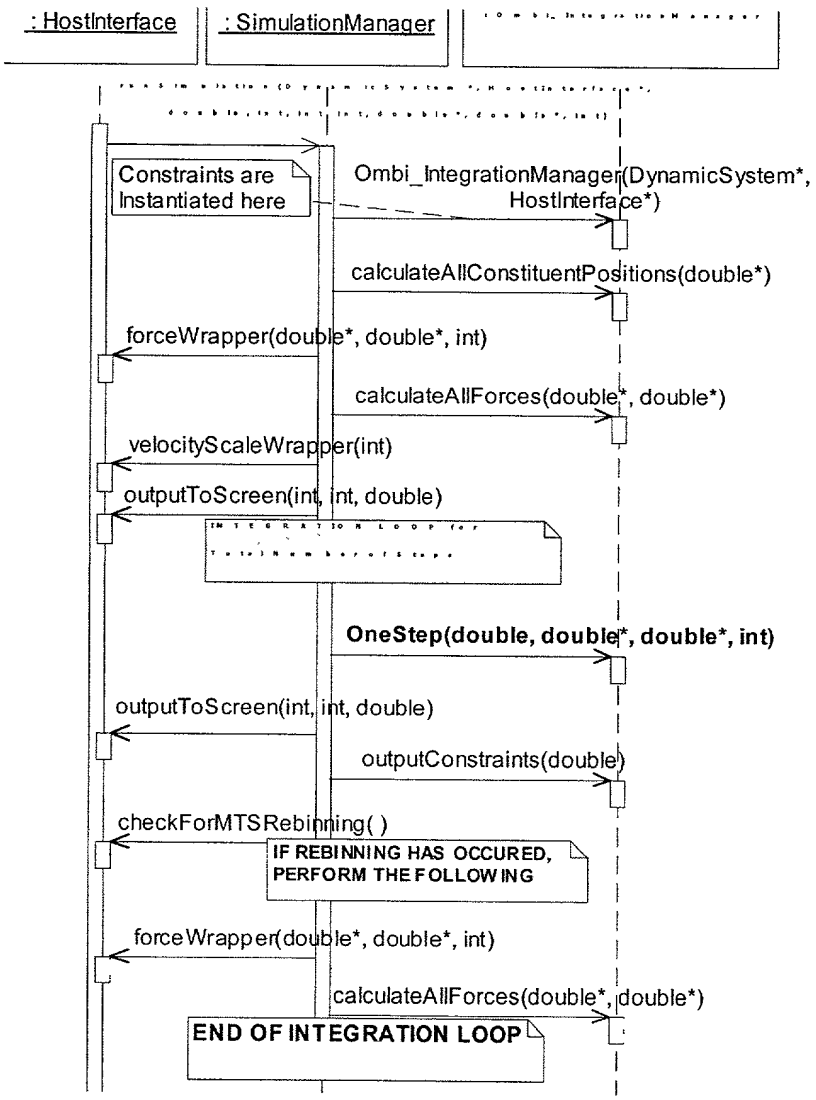


Figure 15. Sequence Diagram for HostInterface::runSimulation Overview

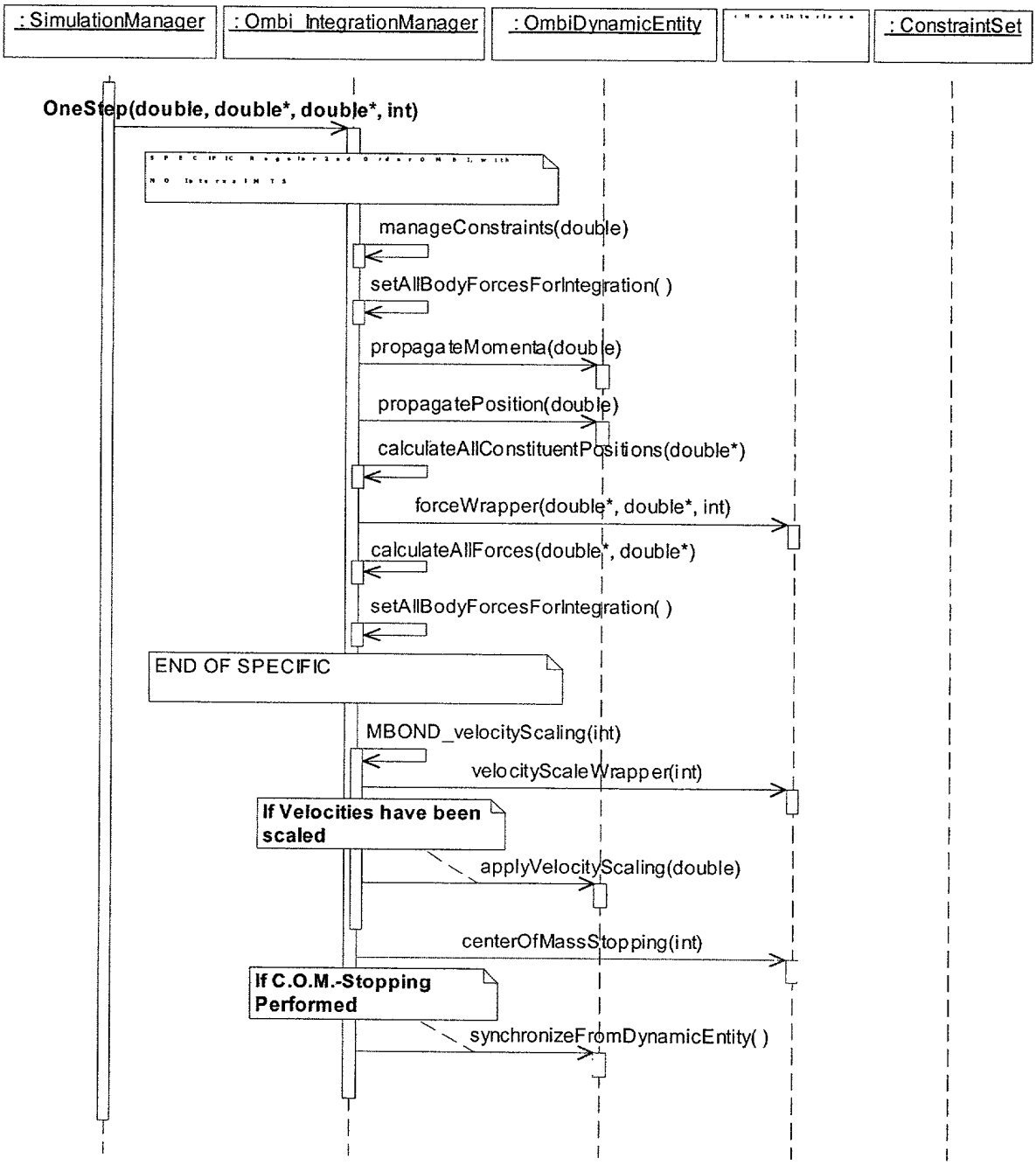


Figure 16. Sequence Diagram One Step of OMBI

8.4.4 Sequence Diagram for Initialization of Constrained Multibody System

The algorithms for initializing the constrained multibody system are given in Section 7.4. Figure 17 and Figure 18 formalize this initialization process via the sequence diagrams. Correspondingly, Figure 17 describes preparations and Figure 18 describes the initialization algorithm itself.

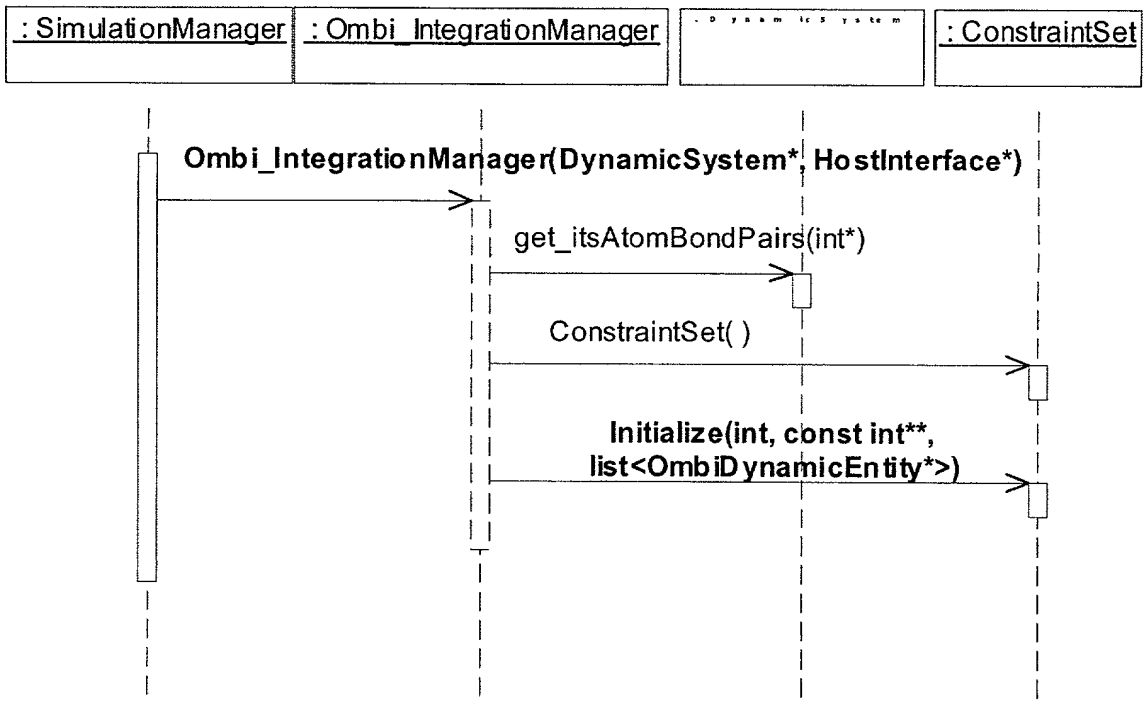


Figure 17. Preparation for Initialization of ConstraintSet

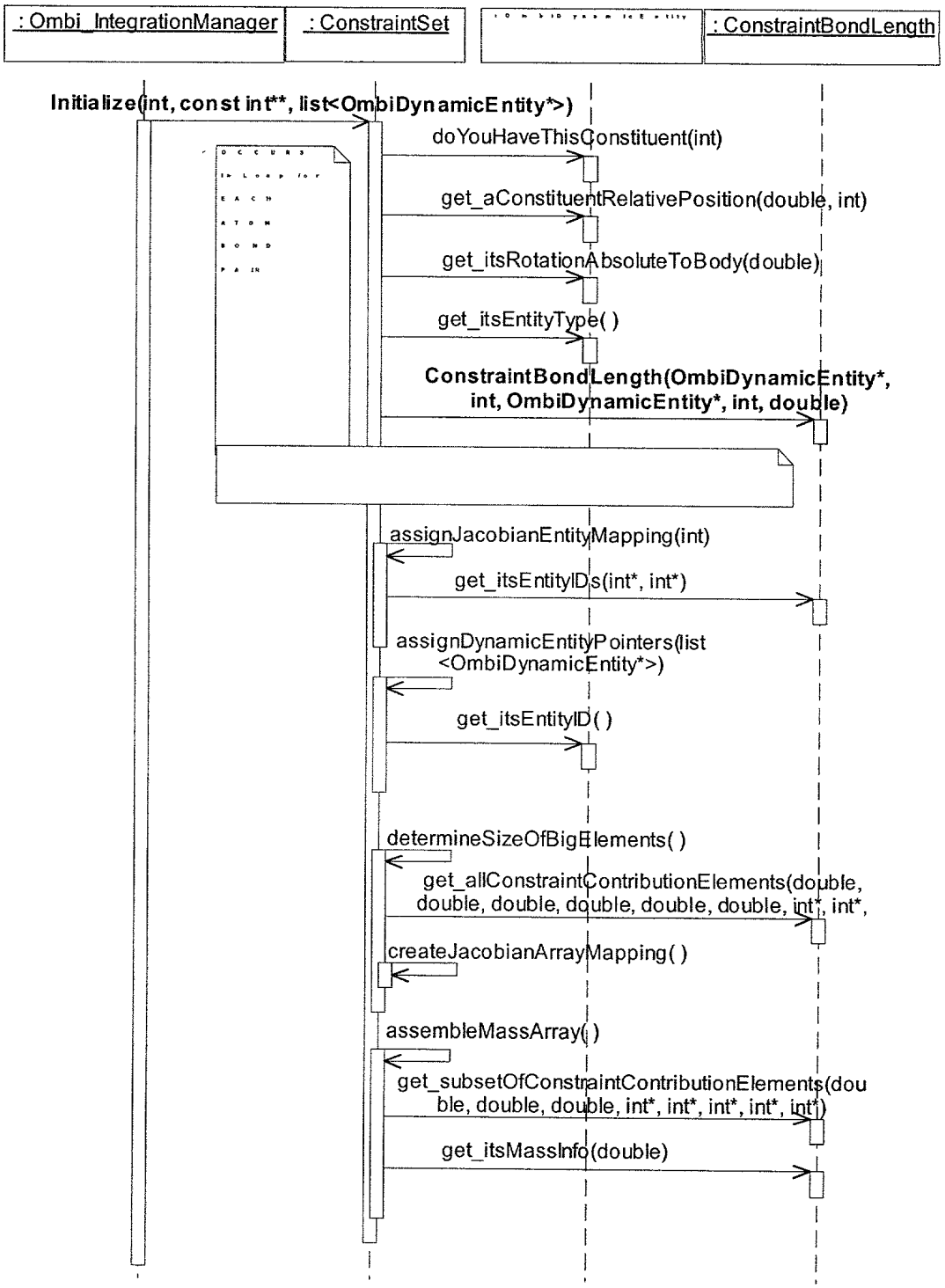


Figure 18. ConstraintSet::Initialize

8.4.5 Sequence Diagram for Integration of Constrained Multibody System

The overview of the OMBI/SHAKE algorithm for dealing with bond-length constraints in multibody MD is presented in Section 7. Figure 19 provides a corresponding sequence diagram for the OMBI/SHAKE algorithm as it is implemented in the OO C++ MBO(N)D-II code. Constraints are incorporated into the OMBI Integration Manager conducting the OneStep(...) operation by means of manageConstraints( ) and setForcesForIntegration( ) operations of Ombi_IntegrationManager, as shown in Figure 19. At the beginning of each step, before the momenta are propagated by the first ½ step, the constraint forces are solved for using iterative SHAKE-type solution. In essence, manageConstraints( ) tells ConstraintSet to solve for the Lagrange multipliers of constrained system. Then, setForcesForIntegration( ) tells ConstraintSet (if there are any constraints) to convert the Lagrange multipliers into constraint forces, and add these forces to the body-forces from force-field, before integrating momenta. Once the momenta have moved ½ step, then the positions are propagated to the full-step and the force-field is evaluated. The constraint forces are functions of the constraint Jacobian, which is dependent upon position coordinates, and therefore we need to compute the constraint forces using a newly computed Jacobian and the same Lagrange multipliers as solved from beginning step.

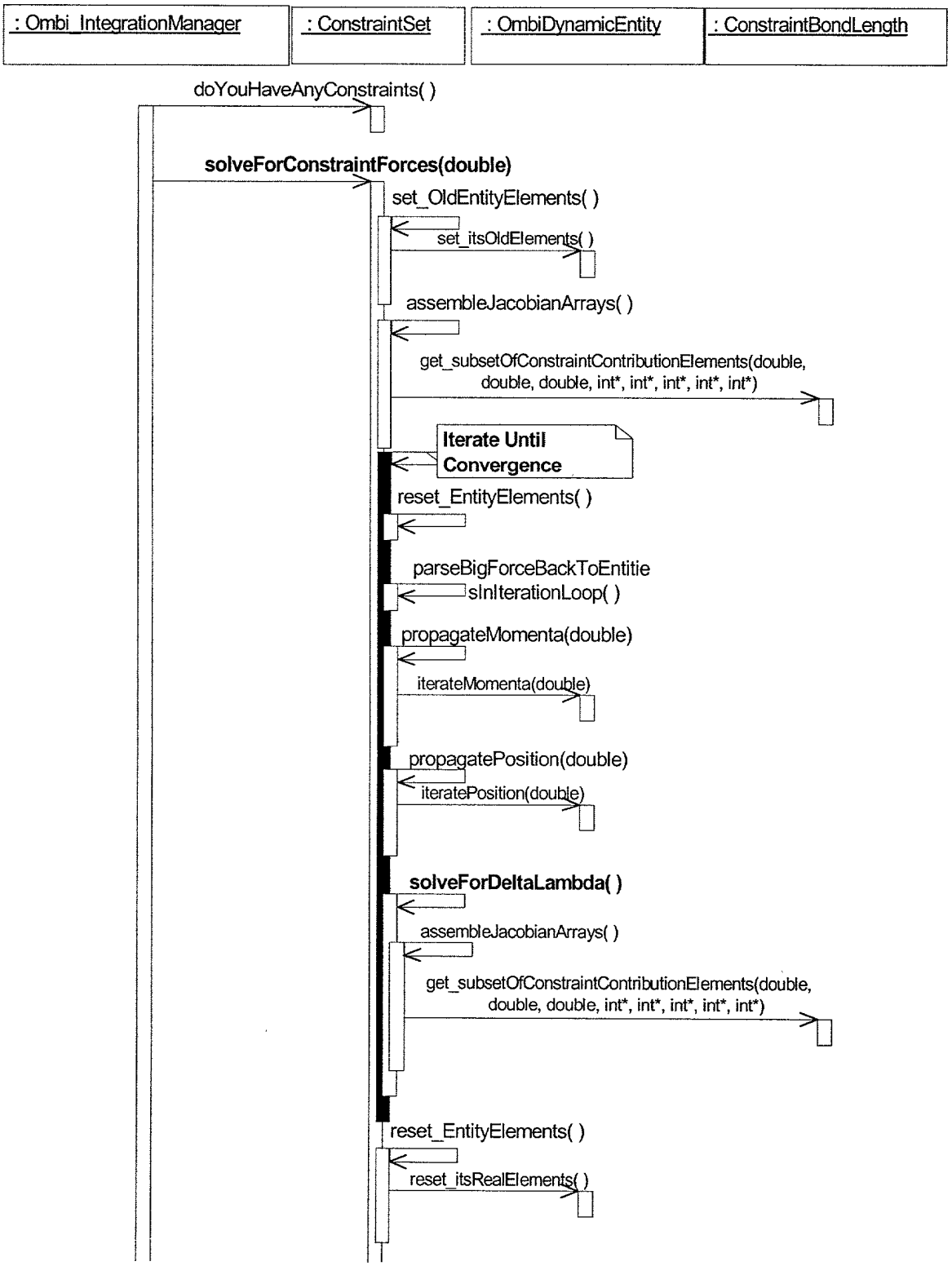


Figure 19. ConstraintSet::solveForConstraintForces(double dt)

9 Task 5: Modify Existing MTSL Strategy to Incorporate OMBI

The integrators introduced in the previous sections including the optimized multibody Integrator (OMBI), are single-time scale (STS) integrators. But, MBO(N)D dynamics are characterized by a diversity of time scales. Modeling groups of atoms as rigid or flexible bodies not only reduces the number of degrees-of-freedom, but also reduces the high-frequency dynamics associated with those atoms. Because some parts of molecules may still be modeled atomistically in the MBO(N)D approach, high-frequency motions are possible. Also, substructuring schemes usually provide bodies of various sizes, which will have different frequency dynamics. Moreover, the frequency of interatomic interactions depends on the distance between atoms, which is larger for long-range interactions and smaller for short-range interactions. For these complex systems with multiple-time scales, using the SST integrator with the time step limited to the highest frequency dynamics within the system is not the best strategy. Therefore, MBO(N)D needs a MTS integrator to maintain its large time step advantage while simultaneously permitting high-frequency motion in energetic regions of the system. The MTS integrators have been implemented (Ref. 60) in molecular dynamics software such as CHARMm [18].

Under the ATP project (Advance Technology Program sponsored by National Institute of Standards), we have already developed an MTS integrator for MBO(N)D. This MTS integrator is based on the Lobatto integrator and is derived following the RESPA/MTS (REference System Propagator Algorithms / Multiple Time Scale) concept developed in Ref. 54. In its turn, the RESPA/MTS concept is a MTS-like generalization of the Liouville-Trotter formalism for deriving time-symmetric integrators. We significantly advanced the RESPA/MTS concept by developing a general *n*-bin MTSL (MTS Lobatto) integrator for systems with velocity-dependent accelerations, i.e. the integrator which samples multibody molecular dynamics using *n* different time steps for evaluation of interatomic interactions. Moreover, we significantly extended the choice of step-size ratios (MTS ratios) while retaining the time-reversibility of the MTSL integrator. We also developed the physical criteria which assess the time scales of interatomic interactions and separate them in different time bins. These criteria are based on the mass and distance factors. At last we developed, the automatic algorithm for determining the appropriate number of time scales and corresponding ratios.

The main goal of the proposed work consisted in generalizing the previous version of the MTS integrator in MBO(N)D-I in order to incorporate the OMBI integrator as a new engine for MTS. The merging of these two significant developments (OMBI and MTS) in multibody dynamics resulted in a powerful numerical strategy integrating multibody molecular dynamics with the large diversity of time scales.

9.1 MTS Criteria

The MBO(N)D substructured modeling approach is ideally suited for treatment by a multiple time scale (MTS) integrator. This is because of the large diversity in body sizes that are possible in this approach – from single atoms that have high frequency content to large bodies that have low frequency content. A natural time scale separation criterion for MBO(N)D is therefore body mass. In addition, a distance dependent criterion is useful for calculating more frequently those nonbond interactions that are of short range, while calculating less frequently those that are of long range.

The MBO(N)D MTS integrator primarily treats the nonbond interactions by sorting the nonbond pair list and placing each interaction pair into a separate bin based on their predicted time scales. Additional multiple time scale separations, such as treatment of bond, angle, and dihedral terms at different time scales, are possible since these are straightforward adaptations from algorithms for atomistic simulations.

The sorting of nonbond interaction pairs into different bins is based on the assumption that the interaction can be characterized as that between two point masses connected by a linear spring. The frequency of such an interaction is

$$\omega = \sqrt{\frac{k(m_1 + m_2)}{m_1 m_2}} \tag{176}$$

where k is the spring stiffness, and m_1, m_2 are the masses. For a given pair of atoms, the stiffness could be approximated by the second partial derivative of the interaction potential. Assuming that electrostatic interactions dominate at longer distances, the stiffness k would have a distance dependency of $1/r^3$. If both atoms were modeled as particles, the masses to be used in the above formula would naturally be the atomic masses. However, if an atom is part of a rigid body, the time scale of the atom's motion would be largely dictated by the mass of the rigid body. We thus define the interaction mass of an atom to be the following: equal to atomic mass if the atom is modeled as a particle or equal to mass of the body if the atom is part of a rigid body or equal to the atomic mass if the atom is part of a flexible body (this is to be conservative).

Incorporating the above considerations, the following equation approximates the appropriate integration step size for each nonbond interaction pair, for atoms i and j:

$$\Delta t_{ij} = C_0 C(r) \sqrt{\frac{M_i M_j}{M_i + M_j}} \tag{177}$$

where Δt_{ij} is the approximated integration step size for the interaction, C_0 is a calibration factor, $C(r)$ represents a distance-dependent interaction compliance factor, and M_i is the interaction mass of the i^{th} atom, as defined above. The calibration factor C_0 puts the relative time scales, determined from ranking the time scales according to the interaction masses, on the absolute time axis. One approach for setting C_0 is to calibrate the time scales with respect to a known interaction. For this purpose, we chose the hydrogen-hydrogen interaction

$$C_0 = \sqrt{2}\, \Delta t_{HH} \tag{178}$$

where Δt_{HH} is 0.5 or 1 fs.

Figure 20 illustrates that the two factors (mass and distance) determine the time-scale of the interaction between two atoms.

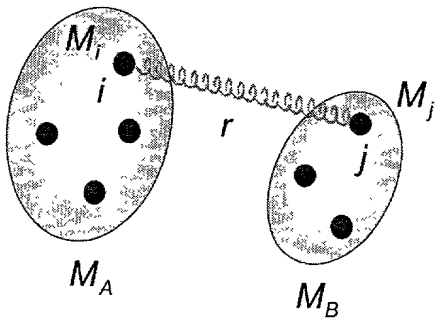


Figure 20. Mass/Distance factors in Estimating Time Scale for Atom-to-Atom Interaction

After performing a number of numerical studies, it was found that the following simple form for the interaction compliance factor is useful,

$$C(r) = \begin{cases} 1 & \text{if } r \le r_0 \\ 1 + A(r - r_0) & \text{if } r > r_0 \end{cases} \qquad (179)$$

The distance dependency in the above expression allows for a distance-insensitive region for short-range interactions. The following choice of parameters has been shown to work well in MBO(N)D test simulations to date: $A = 1 \text{ to } 2$ and $r_0 = 2 - 4\,\mathring{A}$.

Because of the distance dependency of the time-scale separation criterion, care must be taken when updating the nonbond pair list, which would result in changes to the placement of nonbond pairs into the different bins. While many developers of MTS integrators have chosen to implement switching functions to handle this problem [114], a simpler solution of re-calculating the forcefield interactions at the time points at which the nonbond update takes place has been implemented. This approach avoids the use of bin-specific switching functions, which otherwise would become quite cumbersome when applied to more than a small number of time scale bins.

The time scale separation criterion described above has been used to sort nonbond pair lists into an arbitrary number of bins for multiple time scale MBO(N)D simulations. Time scales used have ranged from 1 to 3 fs for short-range interactions between lightweight particles, to 60 to 80 fs for long range interactions between atoms on large heavy bodies. In some cases, 60% to 90% of all the nonbond interactions were sorted into the slowest (80 fs) time bin, resulting in a five- to ten-fold reduction in the CPU time spent on forcefield evaluations.

9.2 Computational Mechanism of Multiple Time Scale Integrator

In order to combine the MTS criterion described above into the integration scheme, a variant of the Lobatto integrator was developed in MBO(N)D-I that allows the evaluation of different forcefield interactions at different time scales. The general framework of applying the Trotter decomposition of the Liouville operator [22] was used to develop a time-reversible multiple time scale integrator. The resulting MTS Lobatto integrator uses weighted combinations of forces from different time scale bins for each velocity update. The weighting

and frequency of calculations from each bin are determined by the time-scale ratios between each bin and the baseline integration time-step.

The Lobatto integrator is very similar to the velocity Verlet integrator, which can be derived from a Trotter decomposition approach. The former can be thought of as an approximate implementation of such an integrator. We will use Trotter decomposition to derive the formulation for the multiple time scale Lobatto integrator. We will derive a three time-scale MTS integrator, with time scale ratios of 1:2:3. The extension of the formulation for additional time scales is obvious. The Liouville operator modified for the position-velocity form instead of position-momentum form is defined by the formula

$$U(t) = \mathbf{e}^{iLt},\ iL = v\frac{\partial}{\partial x} + a\frac{\partial}{\partial v} \tag{180}$$

where t stands for time, x for position, v for velocity, and a for acceleration.

Note that the Lobatto integrator can be thought of as the approximate implementation of the following propagator, derived by Trotter decomposition of the Liouville operator

$$U(\Delta t) = U^{vL}U^{x}U^{vR} \tag{181}$$

where

$$\begin{aligned} U^{vL} &= U^{vR} = \mathbf{e}^{a\frac{\partial}{\partial v}\Delta t/2} \\ U^{x} &= \mathbf{e}^{v\frac{\partial}{\partial x}\Delta t} \end{aligned} \tag{182}$$

The first half-step and second half-step velocity propagation of the Lobatto integrator can be thought of as approximate implementations of the U^{vR} and U^{vL} propagators above.

In this particular example, the acceleration is decomposed into three time scale components

$$a(t) = a_1(t) + a_2(t) + a_3(t) \tag{183}$$

where the index indicates the time scale ratio for evaluation of the acceleration components. The Liouville operator for propagation over a complete integration cycle, defined as the least common multiple of the step sizes of each component can be decomposed into a symmetric propagator as follows.

$$U(6\Delta t) = U^{*}U^{*T} \tag{184}$$

where

$$U^{*} = U_3^{vL}U_2^{vL}\left(U_1^{vL}U^{x}U_1^{vR}\right)^2 U_2^{vR}U_2^{vL}\left(U_1^{vL}U^{x}U_1^{vR}\right)U_3^{v} \tag{185}$$

The superscript T denotes the reverse order of operations in the half-operator U^{*}, with R and L superscripts replacing each other. The component propagators are defined as

$$\begin{aligned} U_k^{vL} &= \mathbf{e}^{\frac{1}{2}(k\Delta t a_k)\frac{\partial}{\partial v}} = U_k^{vR} \\ U^{x} &= \mathbf{e}^{\Delta t v\frac{\partial}{\partial x}} \end{aligned} \tag{186}$$

Direct implementation of the propagator would result in an integrator that had many velocity updates for each baseline integration step, each involving accelerations for the different nonbond bins. A simple modification of the grouping shown below can significantly reduce the complexity of the integrator. The new grouping results in the following propagator.

$$U(6\Delta t) = \left(U_{321}^{vL} U^{x} U_{1}^{vR}\right)\left(U_{1}^{vL} U^{x} U_{12}^{vR}\right)\left(U_{21}^{vL} U^{x} U_{13}^{vR}\right) \cdot \left(U_{31}^{vL} U^{x} U_{12}^{vR}\right)\left(U_{21}^{vL} U^{x} U_{1}^{vR}\right)\left(U_{1}^{vL} U^{x} U_{123}^{vR}\right) \quad (187)$$

where

$$U_{mn}^{v} = \mathbf{e}^{1/2\Delta t(m a_m + n a_n)\frac{\partial}{\partial v}} \quad (188)$$

represents a combined velocity propagation. Note that over the $6\Delta t$ integration cycle, the integrator performs six integration steps, using the baseline step size for each of the position updates.

The difference from step to step lies in the treatment of the velocity half-step propagation. Instead of propagating the velocity several times for each baseline time step, once for each nonbond bin, the new integrator propagates the velocities using a weighted combination of the forcefield terms from all of the relevant bins. The inclusion of forcefield terms from a given bin depends on whether or not the current time point is an integer multiple of the bin's step size. For the velocity propagation involving U^{vL}, the relevant time point is the beginning of the step; for the velocity propagation involving U^{vR}, the relevant time point is the end of the step. For example, the integrator uses forces from the 1 fs and 3 fs nonbond bins, assuming a 1fs base step size, at the $3\Delta t$ time point for completing the $3\Delta t$ interval and starting the $4\Delta t$ interval. Forces from the 2 fs bin are not used at that point.Figure 21 illustrates this scheme for the MTS with 1:2:3 ratios. In particular, Figure 21 shows why a scheme based on time-symmetric force weighting is more general but much simpler for understanding and implementation than e.g. multiple-stage scheme in RESPA [22]. Indeed, the scheme of Figure 21 makes it possible to use the binning with arbitrary integer ratios while RESPA deals only with binning where smaller time steps are embedded into larger time steps (e.g. like in the ratios 1:2:4:8). In the scheme of Figure 21, the full MTS step corresponds to the least common multiple of MTS ratios, i.e. Full Step = Six Base Steps (6 = 1·2 ·3). In this general (but simple) scheme one just needs to use an ordinary single-step integrator while computing the total force at each step via a time-symmetric weighting of forces (F) by their MTS ratios in different bins. The RESPA-type MTS integrator would involve a much more complicated algorithm which uses a time-symmetric sequence of so-called stages. At each stage the velocity state is updated in response to each the force in each MTS bin. In particular, the latter update introduces not only more complex logic but also is less efficient since the dynamics need to be computed as many times as the number of MTS ratios. In the scheme of Figure 21 the dynamics are updated only once since the total force is computed via the time-symmetric weighting of forces.

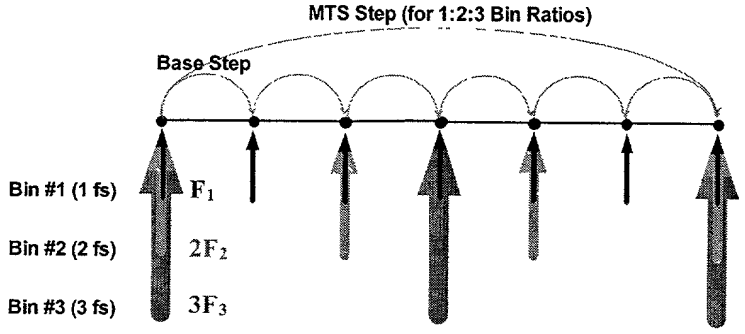


Figure 21. Time-Symmetric Scheme for Calculating Forces in MTS

The particular integrator described above can be easily generalized to MTS integrators with an arbitrary number of bins and arbitrary time scale ratios. This has been accomplished in MBO(N)D-II using the OMBI as a single-step propagator (with the symmetrically-weighted forces).

9.3 Advantages of the MBO(N)D MTS over the CHARMm MTS

Section 9.1 describes the physical criteria for MTS binning used in the MBO(N)D-I and MBO(N)D-II. Section 9.2 describes the computational mechanism of the MTS integrator. The both factors (the physical criteria for binning and the computational mechanism of integration) provide important advantages over the state-of-the-art approaches.

Figure 22 summaizes and illustrates these advantages in a graphical form, using CHARMm MTS [68] as an example. First, the MBO(N)D MTS has a more selective separation of the interacting pairs of atoms into the MTS bins since it uses a combined mass/distance criterion (Section 9.1). The CHARMm MTS [68] uses either distance or mass criterion. Second, as was discussed in Section 9.2, the time-symmetric force weighting in the MBO(N)D MTS is far superior that the computational mechanism of RESPA, implemented in CHARMm. Due to its more powerful computational scheme, the MBO(N)D MTS effectively utilizes a variety of the multiple time scales in MD. For example, as shown in Figure 22, it can use the MTS ratios 1:2:3:4:6:8, while the CHARMm MTS can use its best alternative 1:2:8. Note that the CHARMm MTS has a limit on the maximum number of MTS bin (it equal to 3). The latter limit is a result of the RESPA's complexity, namely due to the multiple-stage archetecture of its computational mechanism for the velocity update. Indeed, the latter requires that each MTS bin has its own list of neighbor atoms (non-bond list). For large systems, mainaining multiple non-bond lists becomes prohibitive due to memory limitations. At the same timer, the time-symmetric force weighting in the MBO(N)D MTS offers an excellent alternative by making it possible to do with a single non-bond list for an arbitrary number of the MTS bins.

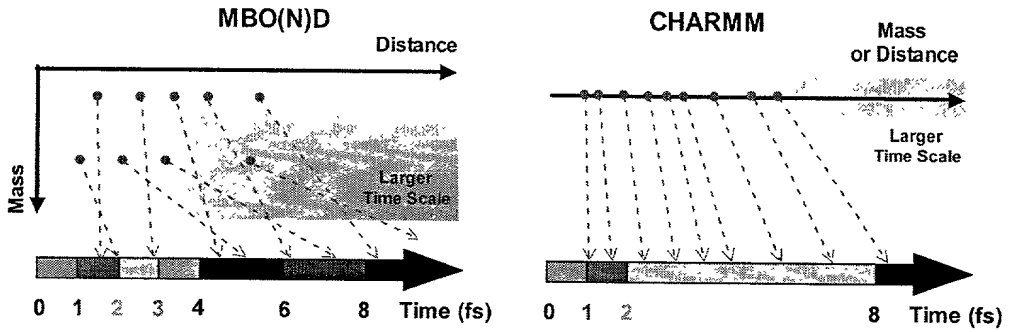


Figure 22. Advantages of the MBO(N)D MTS over the CHARMm MTS

9.4 Incorporating MTS into OMBI

This section describes how the MTS (Multiple Time-Scale) capability is added to the new OO C++ MBO(N)D-II code, which is documented in Section 8. Briefly, the type of MTS we are concerned with allows for sampling of atomic forces at different frequencies. We would like to be able to sample atomic interactions that fluctuate at high frequencies with small time steps, and those interactions that fluctuate at lower frequencies with larger time steps. With non-MTS integration, we are basically left with two choices:

1. For Accuracy, compute all interactions at the smallest time step, thus increasing computational cost, or
2. For Speed, compute all interactions at a larger time step, losing information about the higher frequency interactions, thus decreasing accuracy

MTS allows us to decrease the computational cost (increase speed) while maintaining accuracy. The FORTRAN MBO(N)D code has available to it an MTS routine. On the CHARMm side of the code, if MTS is selected, an analysis is performed on the bond interactions to determine a "binning" scheme for MTS. It specifies the number of MTS "stages", the ratios (multiples of the smallest time step – larger ratios for lower-frequency interactions), and it handles the logic for atomic force calculations for appropriate interactions.

Below we describe the methodology used in the FORTRAN MBO(N)D-I version of MTS, from which we can deduce how to proceed in MBO(N)D-II.

For MTS, rebinning of forces must be accounted for to preserve symmetry of the integration steps. Lets say that the MTS bins are at 1, 5 and 10 fs steps. Then the meaningful output occurs at every 10 integration steps. If rebinning occurs after one such set of 10 steps, the atomic forces will be different. The Lobatto type integrators rely on the forces at the end of the previous step being used at the beginning of the next step. However, after rebinning the required symmetry will not be in place. Therefore, whenever a rebinning occurs, we must recompute the forces for the beginning of the next step. In MBO(N)D-I, the routine lobto(..) checks for rebinning at the end of a step. If rebinning has occurred, then it recomputes the forces. We need to do the same thing for the MBO(N)D-II code.

9.4.1 MTS Procedure in MBO(N)D-I

Further study of the code needs to be performed in order to represent the initialization of the MTS stages and the logic for computing MTS atomic forces. Here, we assume that all of the MTS initialization has been performed correctly on the CHARMm side, and look at how FORTRAN MBO(N)D uses this information to compute atomic forces during integration.

During integration, the FORTRAN function **extor_CHARMm**(...) is called to compute the external forces and torques acting on the bodies. This routine calls **mbforce**(...) to get atomic forces, and then calls **Lforce**(...) to convert atomic forces to body forces and torques. MTS ratios (if there is more than 1 MTS Stage) are found in the COMMON array variable **mratio**, which is copied from **Iparam**[29,30,...] for MTS stages 2,3,... . The number of MTS stages (sometimes referred to in the FORTRAN code as **nstages**) is found in **Iparam** [28]. The current MTS stage (used for the computation of atomic forces associated with a given MTS stage) is set in **Iparam** [17].

The FORTRAN MBO(N)D Algorithm is described below.

STEP 1: Calls to mbforce(...) and Lforce(...) for MTS stage 1

1.1: Set **Iparam**[17] = 1, initializing to the first MTS stage. Apparently, **mbforce** (and who knows what other routines) uses this information.

1.2: Call **mbforce**(...) to get the atomic forces

1.3: Call **Lforce**(...) to convert atomic forces to body forces and torques. The body forces are placed in an array called **GPOS**.

STEP 2: Loop through the number of MTS stages, performing stage-appropriate calculations.

STEP 3: Set **nstages** = **Iparam**[28], the number of MTS Stages.

STEP 4: LOOP through through the number of stages, from 2 to nstages. Note, if nstages = 1, then this step is bypassed. NOTE: The idea is to compute the total body forces as a summation of the Body Forces at each stage, weighted by the MTS ratio: GPOS_total = GPOS_1 + mratio(2)*GPOS_2 + mratio(3)*GPOS_3 + …

4.1: LOOP through through the number of stages, from 2 to nstages. Note, if nstages = 1, then this step is bypassed. NOTE: The idea is to compute the total body forces as a summation of the Body Forces at each stage, weighted by the MTS ratio: GPOS_total = GPOS_1 + mratio(2)*GPOS_2 + mratio(3)*GPOS_3 + …

DO istage = 2,nstages

4.1.a: Set **mrt** = **mratio**(**istage**).

4.1.b: Check if the **Current Integration Step** is a multiple of **mrt**. IF it is, then:

IF (mod(current_step,mrt) .EQ. 0) THEN

4.1.b.1: Set **Iparam**[17] = **istage**, necessary for **mbforce**(..) to compute the correct atomic interactions.

4.1.b.2: Call **mbforce**(…) to get array of atomic forces (the entire array, but some may be zero).

4.1.b.3: Call **Lforce**(…) to convert atomic forces to body forces. Place the body forces in a temporary array called **GPOS_STAGE**.

4.1.b.4: Add these STAGE-specific forces to **GPOS**.

*GPOS = GPOS + **mrt***GPOS_STAGE*

ENDIF

ENDDO

9.4.2 MTS Procedure in MBO(N)D-II

For MBO(N)D-II, we would like to perform any and all CHARMm-specific operations in **HostInterface**. As it stands now, only **HostInterface** knows about **Iparam** and **Rparam**. Only Host Interface should know about MTS as well. Specifically, we would like all of the MTS logic for force calculations to be performed when HostInterface::**forceWrapper**(…) is called. For the AFOSR algorithm, we propose to sum all of the various weighted MTS stage forces at the Atomic level, before the atomic forces are passed back to be converted to Body forces.

Algorithm for the MTS force computations in MBO(N)D-II is described below. The MTS algorithm starts with a call to forceWrapper(…). We add an additional variable to the function elements – theIntegrationStep – for use in determining which MTS stages to include.

STEP 1: Set the number of MTS stages from **itsPtrIparam**:

*int **theNumberOfMtsStages** = **itsPtrIparam[MB_Iparam_NumMtsStages]**;*

where **MB_Iparam_NumMtsStages** = 28-1.

STEP 2: Perform the default pass for atomic force calculations. We need to set the current MTS stage equal to 1 (this is the default for no MTS and the 1st stage for MTS, which represents the forces sampled at every step):

tsIparam[MB_Iparam_MTS_CurrentStage] = 1;

where **MB_Iparam_MTS_CurrentStage =** 17-1.

STEP 3: call **mbforce**(...,**thePtrAtomisticForces**,...)

STEP 4: create a temporary array to hold Atomic Forces if **theNumberOfMtsStages** > 1:

if (theNumberOfMtsStages>1)

*double** ***stagePtrAtomisticForces*** *= new double[itsNumberOfAtoms*3]*

STEP 5: Loop through the MTS stages. If there is only one stage (no MTS) then this routine will be skipped.

for (int ***theMtsStage*** *= 2;* ***theMtsStage*** *<=* ***theNumberOfMtsStages; theMtsStage****++)*

5.1: Get the MTS ratio from itsPtrIparam:

int theMtsRatio = itsPtrIparam[MB_Iparam_MTS_Stage2Ratio+(theMtsStage-2)];

where **MB_Iparam_MTS_Stage2Ratio** = 29-1.

5.2: Check to determine whether **theIntegrationStep** is a multiple of **theMtsRatio**:

if (mod(theIntegrationStep,theMtsRatio) = = 0) {

5.2.a: Update **itsPtrIparam[MB_Iparam_MTS_CurrentStage]** to the MTS stage so that mbforce(...) will know which atomic interactions to use for force calculations:

itsPtrIparam[MB_Iparam_MTS_CurrentStage] *= theMtsStage;*

5.2.b: call **mbforce**(...,**stagePtrAtomisticForces**,...) to assign atomistic forces to the temporary array.

5.2.c: Loop through three times the number of atoms, and add the weighted stage-specific Atomistic forces to thePtrAtomisticForces:

*for (int i = 0; i<3*itsNumberOfAtoms; i++)*

*thePtrAtomisticForces[i] += stagePtrAtomisticForces[i]*theMtsRatio;*

STEP 6: If theNumberOfMtsStages > 1, then delete the temporary dynamically allocated array:

delete[] stagePtrAtomisticForces;

Note that it would appear as though the MBO(N)D-I method would require fewer calculations, because the summations of weighted MTS forces occurs at the body level. To sum atomic forces, we need to loop through 3 times the number of atoms with multiplications. To sum body forces, we need to, at most, loop through 6 times the number of bodies, where generally the number of bodies is much less than the number of atoms. However, this is misleading, because to compute body forces, we still need to loop through all of the atomic forces (in some shape or form), performing matrix multiplications, etc., to convert to body forces. It should be much faster to sum up atomic forces when appropriate for MTS, and only then pass back the Atomic Force array for conversion to body forces.

This is fortunate, because as a goal, we would like to keep all knowledge of MBO(N)D out of as much of the MBO(N)D-II code as possible (which means keeping it in HostInterface). The alternative, computing body forces from each MTS stage and then summing up the

weighted body forces, would require some major revisions to classes such as DynamicEntity, IntegrationManager, etc.

10 Molecular Dynamics Simulations with MBO(N)D-II

This section provides a part of the simulations results which clearly demonstrate that by developing the new dynamics engine for MBO(N)D (OMBI with MTS and SHAKE) we were able to dramatically improve the MBO(N)'s performance. The improvement includes both quantitative improvements (in speed) as well as qualitative improvements such as ability to handle large molecular systems and systems with large amount of water molecules. To show this we compare the MBO(N)D-II (new dynamics with OMBI plus MTS and SHAKE) with MBO(N)D-I (O(N) dynamics with Euler angles plus MTS). In all cases the comparisons with the state-of-the-art molecular dynamics code CHARMm [18] are performed.

10.1 1CTF Molecule

In these 1CTF MD simulations, we demonstrate how MBO(N)D-II significantly improves MBO(N)D-I in terms of CPU time used (even for a relatively small 1CTF molecule with no explicit waters). It is important to note that the MBO(N)D-I simulations were obtained during the ATP project [65] and published in Ref. 5. So, the new simulations were repeated but using now a much better multibody dynamics, i.e. MBO(N)D-II with the OMBI integrator, including also its OMBI/MTS and OMBI/SHAKE generalizations.

The following substructuring schemes were used for the multibody MD simulations. For each scheme, the total number of DOF is listed below, together with the maximum time step used. The atomistic model contained 1185 DOF.

Case 1: Loops and β–strands were substructured into several small rigid bodies with hinges defined at selected ψ dihedrals. Each helix was substructured as a single rigid body. Total of 20 bodies. This scheme is shown in Figure 23. Total number of DOF = 95. Maximum time step = 20 fs.

Case 2: Same substructuring as in Case 1, with body-based modes added to each of the 20 bodies. The modes in each body were sorted by delocalization factor [5] and the lowest 10 modes were added to each helix body. Similarly, the lowest three modes were added to each of the small bodies. 58 modes were added. Total number of DOF = 153. Maximum time step = 10 fs.

Case 3: Entire system was substructured into 31 small rigid bodies with hinges at ϕ or ψ angles. Total number of DOF = 150. Maximum time step = 15 fs.

Case 4: Based on Case 1 substructuring, with each helix further separated into a flexible main-chain body and several side-chain bodies (except for Ala and Gly residues) resulting in 66 bodies. Body based modes with a natural frequency less than 100 cm^{-1} were added to bodies with more than 10 atoms. 107 modes were added. Total number of DOF = 432. Maximum time step = 5 fs.

Case 5: Peptide-planes and side-chains modeled as bodies and C_α atoms as particles. The five lowest frequency modes were added to each side-chain body. Total of 150 bodies and 180 modes. Total number of DOF = 966. Maximum time step = 1 fs.

Case 6: Domain-based substructuring. Two small domains in 1CTF were separated into sub-domains including: $\alpha\alpha$ domain (which includes helices A and B), and the B-sheet

domain (which includes the β–sheet and helix C). Those domains were substructured into bodies, which had several residues each. Then, linker residues, connecting between the two domains, were substructured with 1-residue bodies. Total of 20 bodies. Total number of DOF = 95. Maximum time step = 20 fs.

In these substructuring strategies, Case 1 is the coarsest substructure scheme, and Case 5 is the finest.

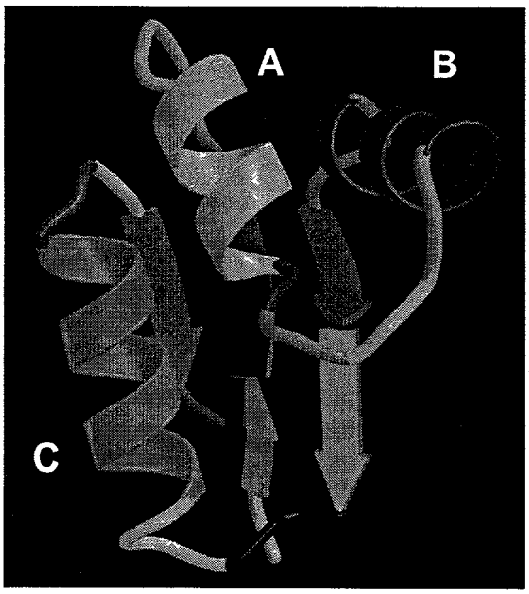


Figure 23. Structure of the C-terminal fragment of the ribosomal protein from E. Coli, (1CTF). The three helices are marked as A, B, and C. Substructured according to Case 1; the residue ID numbers between bodies are shown.

We performed additional simulations using the MTS integrator for Cases 1, 2, 3, and 6 to evaluate the performance of this methodology with different substructuring. MTS was not utilized with Cases 4 and 5 as the potential for simulation speed-up was in the range of that attainable by atomistic techniques.

We explored different values of integration time-steps and MTS ratios to determine the relationship between each different substructuring scheme and stability of the simulation. The conservation of total energy was monitored in order to evaluate the stability of each simulation (100 ps in length). We will only discuss results of simulations exhibiting the highest time step that resulted in stable simulations (defined as $\leq$ 1 kcal/mol RMS fluctuation of the total energy).

Figure 24 shows the speedup versus the ratio of the MBO(N)D-I to atomistic B-C angle RMS fluctuation. The substructuring cases are marked by numbers from 1 to 6. Note that a more elaborate nomenclature for the run is provided which shows a number of loops, bodies and modes in a substructuring scheme as well as the base timestep. Note that a case may have two runs: one with a single timestep (STS) and another with multiple timesteps (MTS). As expected, Figure 24 shows that finer MBO(N)D-I substructuring leads to better agreement with the atomistic results.

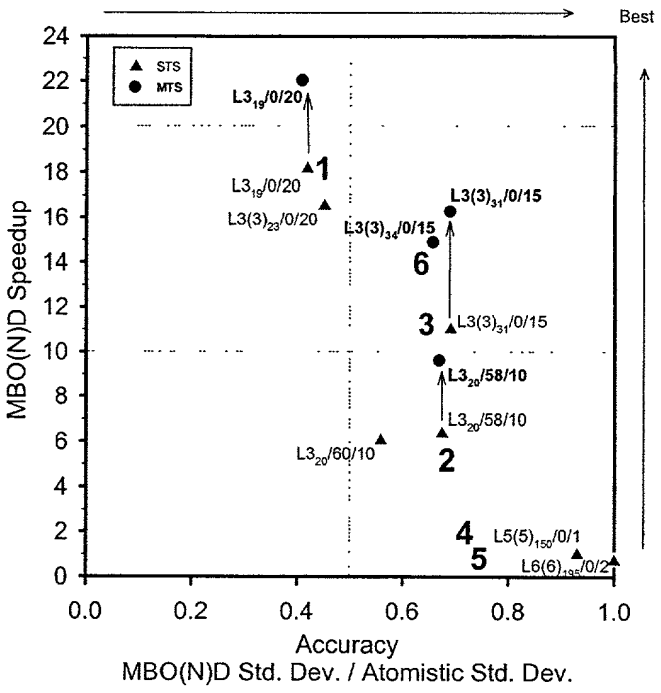


Figure 24. Speedup versus ratio of MBO(N)D-I to atomistic normalized standard deviation of the B-C interhelical angle.

Figure 24 also shows that the MTS integrator increased the CPU speedup over the single-time-scale integrator by a factor of 1.2 to 1.6 for the 1CTF simulations. Comparison of the accuracy and speedup values for Cases 1 and 2 shows that the inclusion of flexible modes can sometimes result in the need for a smaller integration step size in order to maintain stability. Including modes yields a smaller speedup, but significantly improves the agreement with atomistic motions, illustrating the speed/accuracy trade-off in MBO(N)D simulations. Note that Case 5, which represents a very fine level of substructuring, shows excellent agreement with the atomistic result, albeit with little computational speedup. The domain-based substructuring approach (Case 6) shows the greatest level of speedup (factor of 32 with the MBO(N)D/MTS integrator), and still results in a reasonable level of accuracy. In general, one can observe that the finer substructuring schemes require small integration step sizes (1 fs for Case 5). Conversely, the coarser substructuring schemes can use large integration step sizes (20 fs base step size for Cases 1 and 6). The slowest time scale bins for the MTS runs were updated at intervals that were as long as 80 fs.

Figure 25 provides the 1CTF simulations with MBO(N)D-II similar to the simulations with MBO(N)D-II shown in Figure 24. It is important to note that the new and old MBO(N)D-I simulations are not identical due to the fact that the CHARMm forcefield was modified during the time passed since the simulations of Figure 24 were performed. For this reason we rerun the MBO(N)D-I simulations again and placed them on Figure 25. One can clearly see from Figure 25 that even for a relatively small 1CTF molecule MBO(N)D-II speed-ups MBO(N)D-II by a factor of up to 5x. As was mentioned above this was achieved due to more advanced dynamics with the OMBI integrator with MTS and SHAKE. It should be noted that with a coarser substructuring scheme, the MBO(N)D-II's speed-up factor reduces to 2x (with MTS). This is explained by the fact that a fewer number of bodies are used in the substructuring so that O(N)

dynamics become also fast and somewhat catch-up with those of OMBI. Nevertheless, the factor of 2 in speeding-up the MBO(N)D for coarse substructuring means that the MBO(N)D-II reaches a factor of 60x over CHARMm (Figure 24 compares MBO(N)D-I and CHARMm). However, with a coarser substructuring the accuracy of multibody MD significantly drops (below the 0.5 level in the relative B-C angle as shown both in Figure 24 and Figure 25). That is why we consider that the ability of the MBO(N)D-II to use a finer substructuring (and, thus, to achieve a better accuracy) without the loss of efficiency is very important for MD simulations.

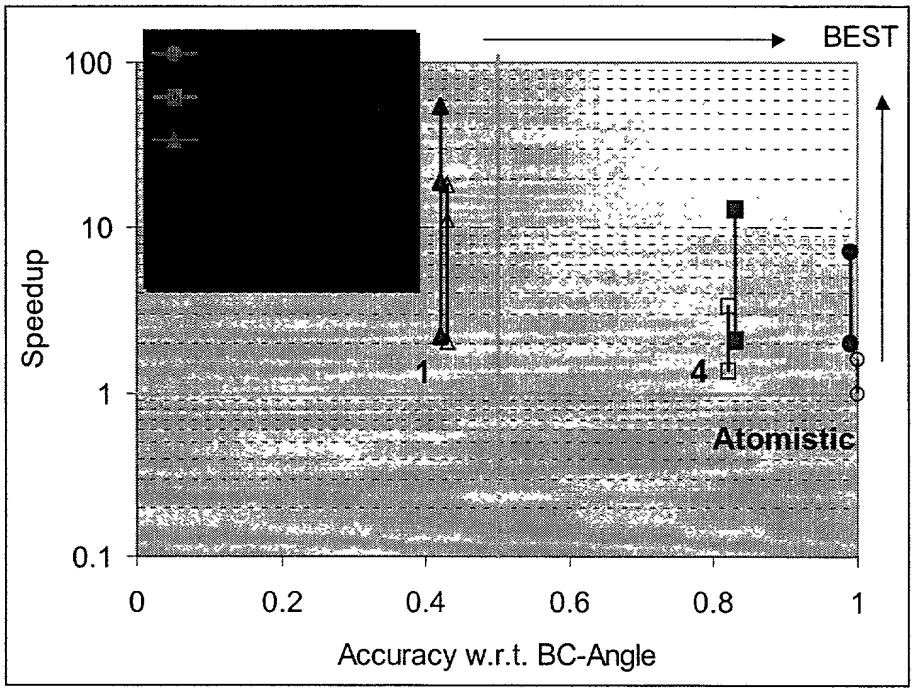


Figure 25. Speedup versus ratio of MBO(N)D-I (MBO(N)D) and MBO(N)D-II (AFOSR) to atomistic normalized standard deviation of the B-C interhelical angle.

Figure 26 provides an important evidence that after implementation of the new dynamics engine (OMBI) in MBO(N)D the accuracy of the MD simulations was retained. In particular, Figure 26a shows the accuracy characteristics in terms of total energy fluctuations. Figure 26b shows the accuracy characteristics in terms of RMS fluctuations of C_α atoms. Figure 26 actually proves that the implementation of the MBO(N)D-II equations is correct. It should be emphasized that this similarity of simulations is an expected fact since the same multi-body scheme (substructuring) is used in both simulations. The differences, however, between MBO(N)D-II and MBO(N)D-I is in the fact that the former provides much faster dynamics and enables solution of large-scale MD.

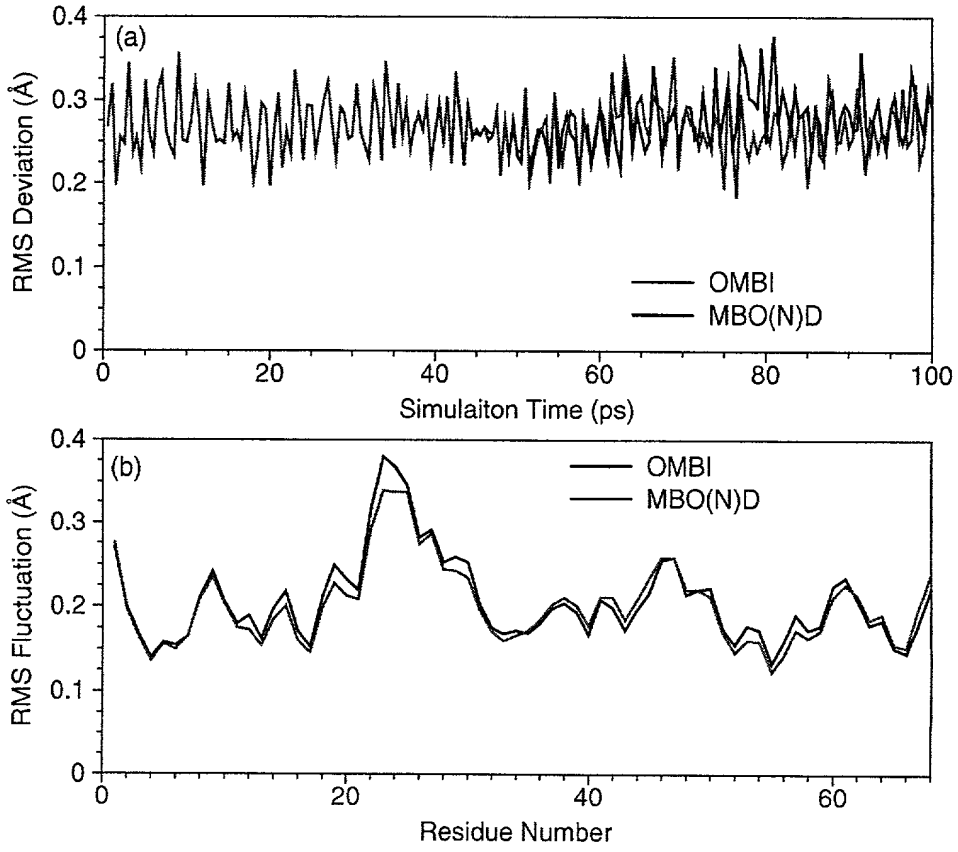


Figure 26. MBO(N)D-II Dynamics Retain the Accuracy of MBO(N)D-I Dynamics when the Same Substructuring is Used

Finally, Figure 27 explains the mechanism of the MBO(N)D-II (AFOSR) superior speed (over MBO(N)D-I or just MBO(N)D as shown in Figure 27). The main explanation is in the fact that in MBO(N)D-I the percentage of the dynamics is 35-80% (the percentage is larger for finer substructuring) and in MBO(N)D-II this percentage drops to a practically negligible level. In other words the OMBI dynamics become very cheap so that they are comparable with the overhead operations in MD (like velocity scaling and computing the data for analysis during the simulation run). The latter fact (that dynamics are very cheap) facilitates unleashing the power of MTS, i.e. computing forcefield interactions at different frequencies. In particular, Figure 27 shows that with MTS (with ratios 1:10:20) the OMBI dynamics still take less than 10%. This makes it possible to consider the forcefield computations as a next bottleneck in speeding-up MD. We plan to address this type of optimization in our further work (beyond this AFOSR project) via implementation of our temporal-spatial decomposition of the forcefield computations.

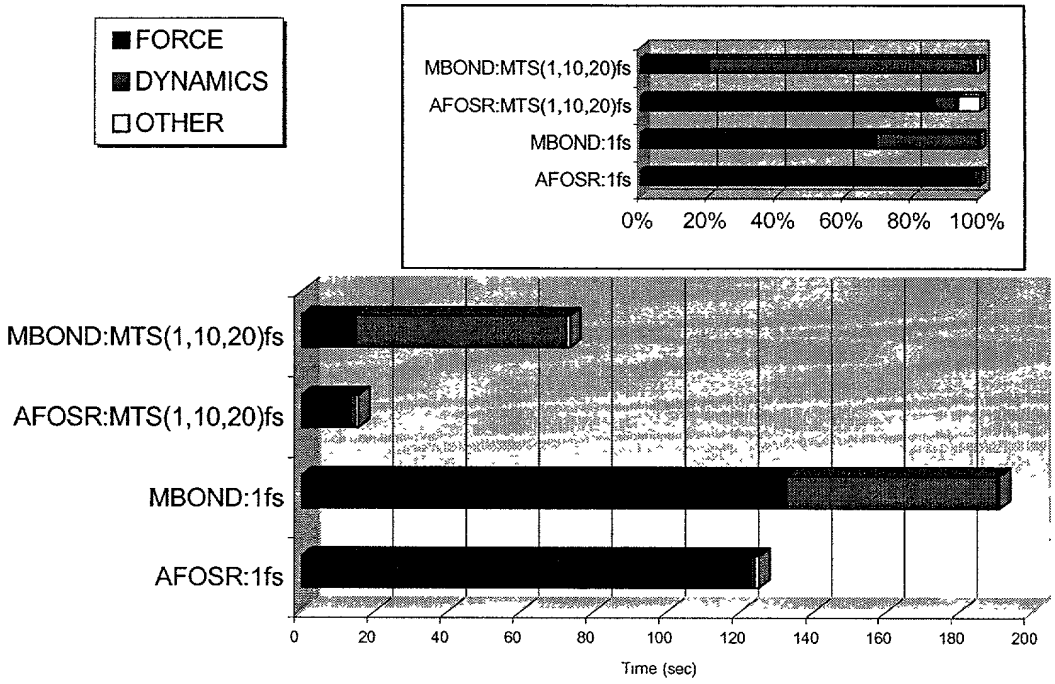


Figure 27. Percentage of Dynamics in MBO(N)D-II

10.2 1CTF Molecule with Explicit Solvation Model

The MD simulations in Section 10.1 were performed under assumption that a 1CTF molecule is not solvated i.e. there are no water molecules modeled either explicitly or implicitly. On the one hand, this type of gas-phase MD simulations allowed us to concentrate on the issues of the substructured MD, as a preliminary step towards studying fully-solvated proteins. On the other hand, the MBO(N)D-I was severely limited in its ability to use a large amount of water molecules. The latter fact was due to use of Euler angle parametrization which in the case of a large amount of small rigid bodies (for each independent water molecule H_2O) caused instability of the dynamics. Also, the memory limitations of the MBO(N)D-I due to the use of the tree structures in the O(N) recursive algorithm prevented the use of large solvated systems.

The development of the OMBI with Euler parameters (quaternions) parametrization resulted into a much better version of MBO(N)D (MBO(N)D-II). This Section presents the MD simulations for 1CTF molecule with MBO(N)D-II which accounts for large number of explicit waters. The direct comparisons between MBO(N)D-II and CHARMm are performed, which demonstrate the advantage of the MBO(N)D-II over CHARMm in this conservative (fully-solvated) MD simulations. It should be emphasized that we call these simulations conservative since we do not sacrifice any accuracy for speed. Correspondingly, the speed-up factor over CHARMm is not 60x (as shown in Section 10.1 for multibody MD with aggressive substructuring scheme) but only ~2x. But, this is considered as a significant improvement of the state of the art in a very competitive area of high-precision solvated MD simulations (in a fact every MD code like CHARMm and AMBER have these "conservative" capability to model a large bulk of explicit water molecules).

To study typical situations of solvated MD simulations, we studied two systems of 1CTF molecule. One system corresponds to a folded state (see Figure 28) and another system corresponds to an extended state (see Figure 29). Consideration of these two systems is important for studying protein folding in a solvent environment (which is the only realistic model of protein folding since in a living cell a newly expressed amino acid sequence is immediately surrounded by water molecules). Note that we do not consider these two systems in the order from the extended to the folded state in order not to give any impression that in this project we attempt to solve a protein folding problem. The protein folding problem is one of the central problem in structural biology. The goal of this AFOSR project is a development of optimal integrators for multibody dynamics. However, we consider that thinking about the challenges of integrating a large-scale dynamic system with thousands of water molecules (modeled as rigid bodies) helped us to solve a long-standing problem in the multibody dynamics such as integrating Euler parameters (quaternions) without the need to artificially enforce the $|e| = 1$ constraint (see Section 6). Although not as fundamental as the protein folding problem, the "quaternion" problem has been attempted by many applied mathematicians during the last 200 years since the invent of the Euler parameters. Again, the challenges of the MD simulations helped to solve the problem of integrating quaternions so that researcher in other applications (aerospace, astronomy, etc.) will benefit from its solution.

Below we describe the solvated MD simulations performed during the Phase II work.

SYSTEM I (folded state):

Folded 1CTF (68 residues), Solvent: a 10A layer of water (735 water molecules)

SIMULATIONS:

CHARMm atomistic runs

MBO(N)D-II runs:

1. Rigid waters, atomistic protein
2. Rigid waters, amide bond placed in a rigid body.
3. Rigid waters, protein represented by 20 bodies.

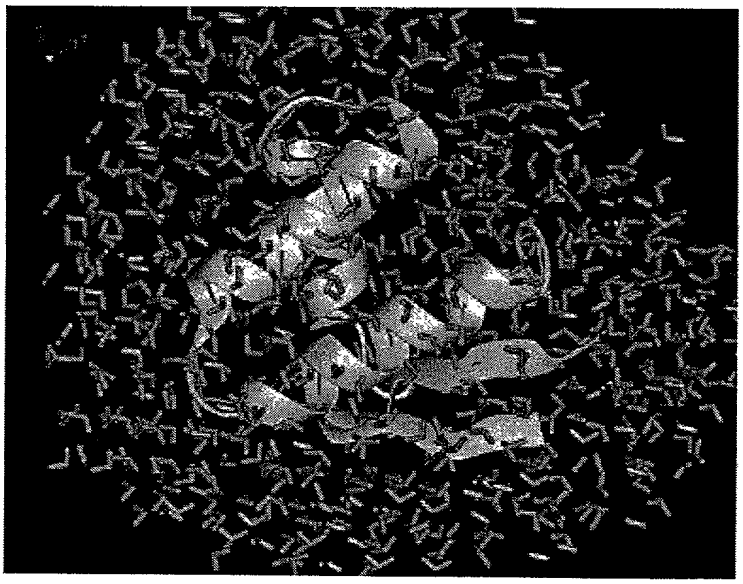


Figure 28. 1CTF Protein Molecule in Folded State with Explicit Solvation Model

SYSTEM II (extended state):

Extended chain 1CTF (68 residues), Solvent: a 8A layer of water (2000 water molecules)

SIMULATIONS:

CHARMm atomistic

MBO(N)D C++ with:

1. Rigid waters, atomistic protein
2. Rigid waters, amide bond placed in a rigid body.

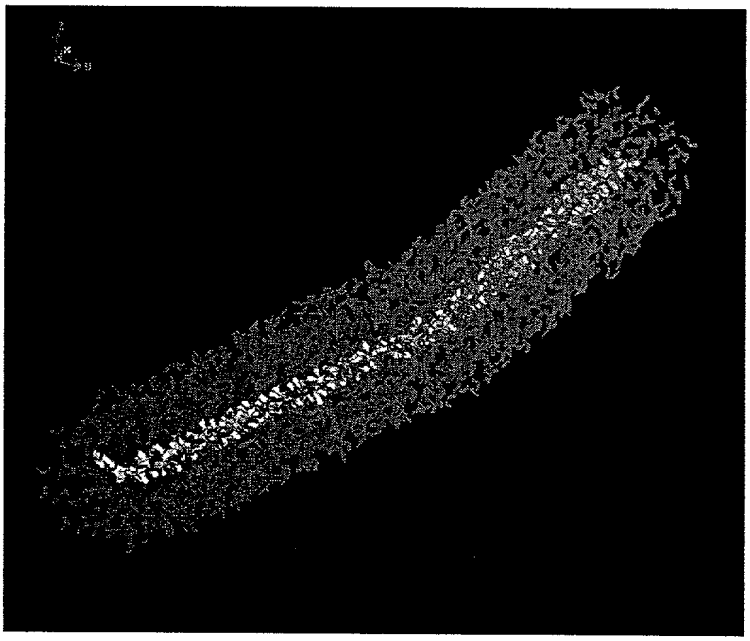


Figure 29. 1CTF Protein Molecule in Extended State with Explicit Solvation Model

Table 4 summarizes the simulation results with MBO(N)D-II (OMBI+MTS) for the 1CTF molecule with the explicit water molecules as specified above. The simulation time was set to 1 ns. Note that we used a CHARMm's option of keeping water molecules around the protein via a harmonic forcefield. Table 4 shows the type of system (folded or extended), a substructuring scheme, the MTS parameters (the bin ratios and a cutoff r_0 in Å specified in the MTS physical criteria of Eq. (179)). The last column in Table 4 provides the speed-up factor of MBO(N)D-II over CHARMm. Note that CHARMm was used with its best MTS version, which includes the distance MTS criteria and the optimal MTS ratios specific for this criterion. As can be seen from Table 4, MBO(N)D-II is faster than CHARMm by a factor of ~2x. Again, this is very important result because we have achieved this speed-up in conservative MD simulations without loss of accuracy. Correspondingly, we have used either fully atomistic model for protein (still having a speed-up of ~1.7x over atomistic CHARMm) or a very fine substructuring scheme based on rigidizing peptide planes (which does not lead to any significant sacrifice of accuracy since the

peptide planes are predominantly rigid in atomistic simulations). As was mentioned above in all MBO(N)D simulations we have used water molecules as rigid bodies. This is a more natural choice than modeling water molecules atomistically. Indeed, even atomistic CHARMm has an option to rigidize the water molecules via a SHAKE constraints algorithm. In these simulations we used the latter option since it provided the best CHARMm's performance.

Table 4. Speed up of MBO(N)D-II vs. atomistic CHARMm (the same accuracy)

SYSTEM	SUBSTRUC.	OPTIMAL BIN RATIO	RCUT	SPEED UP FOLD
Folded CTF in solution	Water as rigid bodies, atomistic protein	3:6:12:24	2.0	1.7
Folded CTF in solution	Water as rigid bodies, atomistic protein	3:6:8:12	2.0	1.7
Folded CTF in solution	Water as rigid bodies, rigid peptide plane	2:3:6:8	2.0	2.0
Folded CTF in solution	Water as rigid bodies, rigid peptide plane	3:8:12:24	2.0	2.0
Folded CTF in solution	Water as rigid bodies, protein represented by 20 bodies	3:8:12:24	2.0	2.1
Unfolded CTF in Solution	Water as rigid bodies, rigid peptide plane	3:8:12:24	2.0	1.8

Note that both MBO(N)D-II and CHARMm use MTS which provides an additional factor of 2-3 over the MD simulations with single time step. Also, note that MBO(N)D-II's MTS will be optimized via further developments which we plan beyond this project. In particular, we envision that through taking a full control of forcefield calculations (i.e. via a separating MBO(N)D from CHARMm) we will be able to implement a better version of MTS (temporal-spatial MTS with switching functions) and achieve an additional factor of 10x and more in speed.

We should clarify that in these simulations we do not have any results from the old MBO(N)D-I code since it could not handle this large amount of explicit water molecules (the maximum number is about 200-300 due to memory limitations and this is for a short simulation time due to instability of Euler angle parametrization). Given the latter limitations of MBO(N)D-II, we consider that by developing MBO(N)D-II in Phase II we made not only quantitative improvement in speed of MBO(N)D but also crucial qualitative changes to enable its application to large solvated systems.

10.3 Myosin S1 Molecule

The goal of these Myosin S1 MD simulations is to show that MBO(N)D-II can handle large biologically important molecules. A Myosin S1 molecule consists of 1158 residues and a total of 11205 atom.

The importance of the Myosin S1 molecule is highlighted below:

- Myosin is a chemo-mechanical transducer responsible for muscle contraction.
- Myosin produces movement by contracting while bound to an actin filament.
- The power stroke mechanism of the myosin-actin system is shown in Figure 30.
- Myosin undergoes very large motion (50 to 100 Å) during the power stroke.
- Myosin S1 head coordinates was solved by a group in Heidelburg.
- The Myosin S1 constitutes the minimum structural information needed to see the putative motion of Myosin
- Myosin S1 is the largest molecule studied by us to date (11200 "atoms" using the united atom force field in CHARMm).

We used the applied force simulation (AFS) method to mimic experimental Optical Tweezers Microscopy (OTM) studies of Myosin S1. The pulling vector, which originates at Phe:835, is shown in Figure 30; the orientation of this vector is similar to that of the OTM experiment. We stretched the molecule, and then released the applied force, thereby allowing the molecule to relax energetically. The length of the simulation was 2.3 ns (which was the length of simulation completed when the quarter ended). The motion of this stretching and relaxation phase was compared to the three lowest frequency normal modes. To mimic the binding of Myosin S1 to the Actin filament, residues 405 – 415, 529 – 550, and 626 - 647 were constrained harmonically about their initial positions. A possible error from the nonbonded cut-off should be also eliminated for this large system. However, MBO(N)D cannot be used with infinite nonbonded potential truncation because of the memory overflow (The source of the memory limitations of MBO(N)D is due to O(N) topological constructions). Therefore, we decided to use the larger cut-off distance than the regular simulation in order to minimize the error from the potential cut-off.

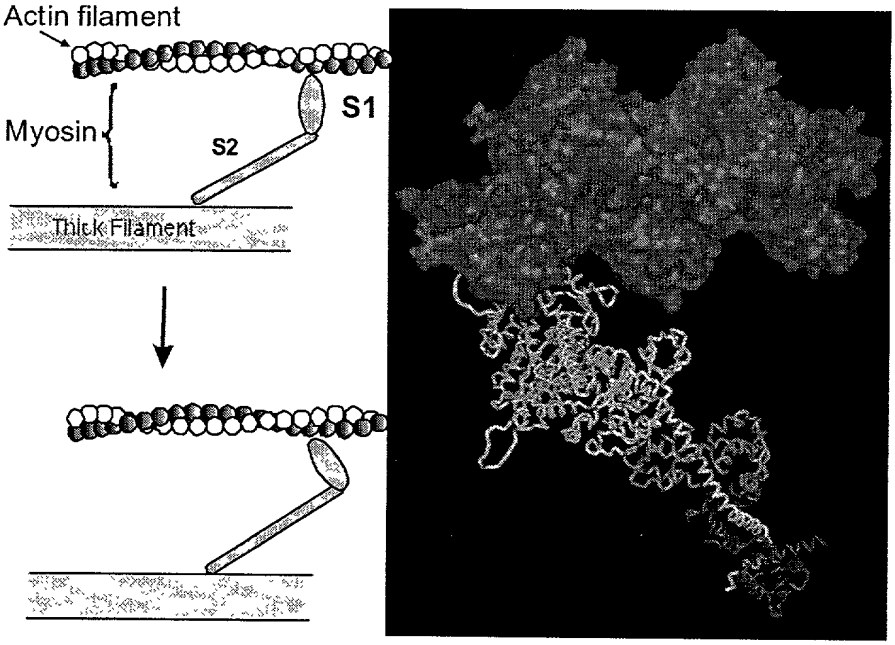


Figure 30. The Myosin/Actin system during the power stroke (left). The Cα trace of the Myosin S1 used in the MBO(N)D simulations (a model of the Actin filament is shown for reference in gray).

The MBO(N)D-I and MBO(N)D-II simulation length for the stretching and relaxation phase was 2.3 ns, and the results compared to the normal mode analysis. The displacement of Myosin S1 during the stretching and relaxation phase was mostly mode 1 (Figure 31a), some mode 3, and very little mode 2. This result is consistent with the fact that the low frequency modes dominate the motions of biomolecules. Figure 31b (the root-mean-square deviation—RMSD—of Phe:835 over time) shows that the stored elastic energy is returned mostly as mechanical work during the relaxation phase. Figure 32 shows the range of motion during stretching as compared with mode 1. The range of motion of Myosin S1 from MBO(N)D was 80 Å. The 2.3 ns MBO(N)D-I simulation was faster than the normal mode calculation by a factor of 5 for a single time step (10 fs), and by a factor of 10 for multiple time step. The absolute lengths of time for the computations are even more impressive: the normal mode analysis calculation required 6 weeks of CPU, while the MBO(N)D-I simulation on the same processor required as little as 4 days.

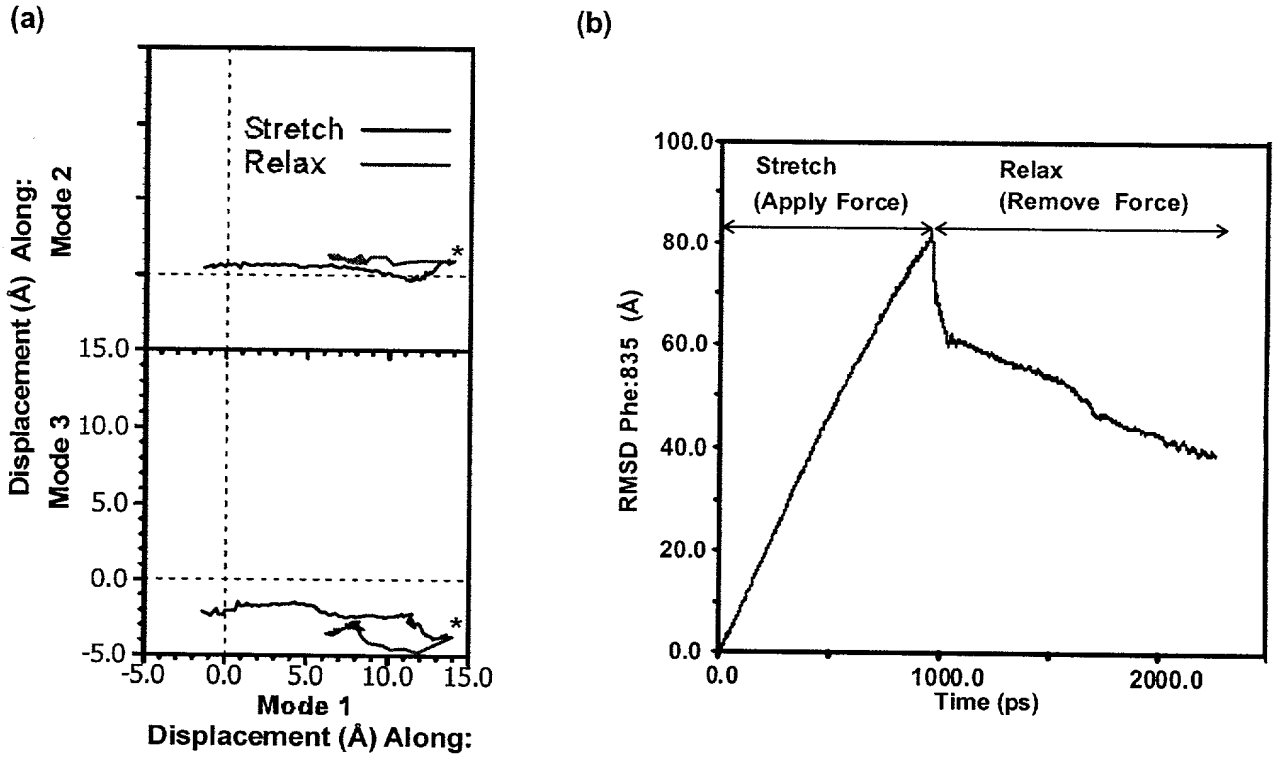


Figure 31. (a) The displacements of the coordinates from the MBO(N)D trajectories in planes defined by mode 1-2 and mode 1-3. (b) The RMSD of Phe:835 over time during the stretching and relaxation phase.

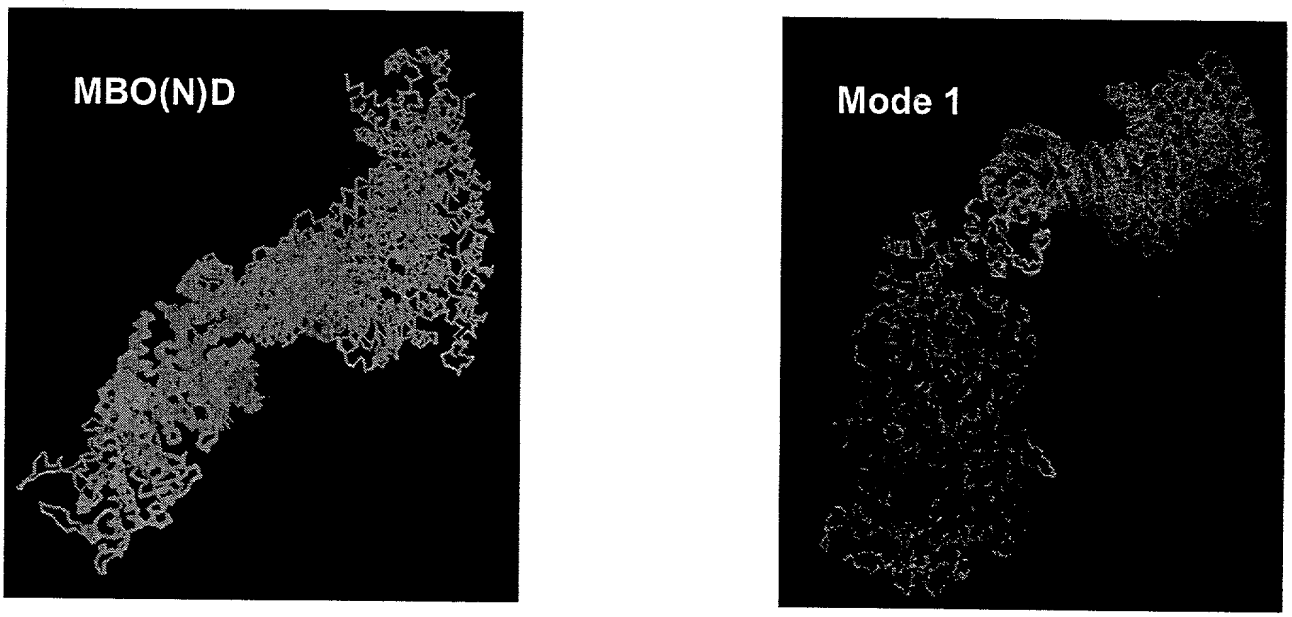


Figure 32. The range of motion and displacements from MBO(N)D (left) and mode 1 (right)

After the Myosin S1 simulations with MBO(N)D-I we made exactly the same simulations with MBO(N)D-II. First of all, we confirmed that with the same substructuring scheme the MBO(N)D-II has the same accuracy. Figure 33 illustrates that the fluctuations of total energy in MBO(N)D-I and MBO(N)D-II are almost identical.

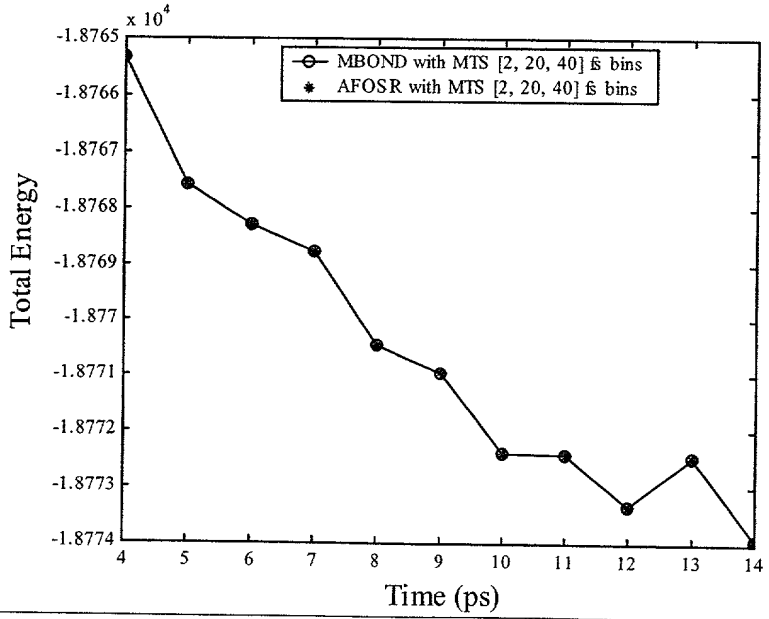


Figure 33. Fluctuation of Total Energy in MBO(N)D-I (MBO(N)D) and MBO(N)D-II (AFOSR)

In terms of speed, we were able to make the new MBO(N)D-II code (AFOSR) faster than the old MBO(N)D-I code (MBO(N)D) as follows

$$\frac{\text{MBOND (2, 20, 40) fs}}{\text{AFOSR (2, 20, 40) fs}} = 1.4$$

Note that the speed-up factor of 1.4x makes MBO(N)D-II faster than the CHARMm atomistic simulations by a factor of 14x. We should clarify that for a smaller 1CTF molecule (see Section 10.1) the MBO(N)D-II's speed-up factor (over MBO(N)D-I) was larger (~2-5x). Does it mean that with a size of molecule the advantage of MBO(N)D-II over MBO(N)D-I becomes less impressive? The answer to this question is very simple. In these particular simulations we used a very coarse substructuring (4 residues per body) thus having only 290 bodies in multibody MD simulations. In this case the percentage of dynamics calculations is only about 5% before MTS and about 40% after MTS (as shown in Figure 34). Correspondingly, the main improvement in the performance come from MTS (about ~3.5x in the overall 10x peed-up factor achieved in MBO(N)D-I over the CHARMm atomistic simulations). The situations dramatically changes if one selects a finer substructuring. Our simulations showed that MBO(N)D-I cannot handle more than 300 bodies due to memory limitations of the O(N) constructions. Still, as our estimates, show if there was enough memory for MBO(N)D-I, the MBO(N)D-II OMBI/MTS simulations would be faster than MBO(N)D-I MTS simulations by ~10 times.

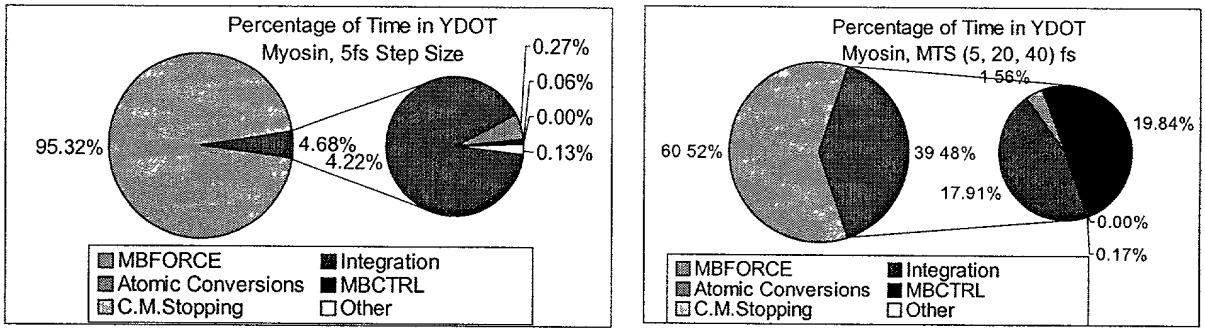


Figure 34. MBO(N)D-I Profiling

We performed additional MBO(N)D-II simulations for the Myosin S1 molecule which indicate that finer substructuring schemes (with 1 residue) may be used without significant loss of speed. This is only purely technological conclusion, which was driven by the goal of this AFOSR project. In other words, by developing MBO(N)D-II we have significantly expanded the application scale of MBO(N)D allowing large-scale molecular simulations. MBO(N)D-II is an excellent tool to study the motion of molecules that are simply too large to be studied efficiently with atomistic methods, on machines other than costly massively parallel supercomputers. This new feature of MBO(N)D-II will be exploited by us and our customers in numerous scientific applications (which are beyond the scope of this technological project).

11 References

1. Amadei A., Linssen A.B.M., and Berendsen H.J.C., *Proteins*, **17**, 412 (1993).
2. Arnold V.I., Kozlov V.V. , and Neishadt A.I., "Mathematical aspects of classical and celestial mechanics", in *Dynamical Systems III, Encyclopaedia of the Mathematical Sciences Volume 3*, ed. Arnold V.I. (Springer-Verlag, New York, 1988).
3. Barth E., Kuczera K., Leimkuhler B., and Skeel R.D., "Algorithms for constrained molecular dynamics", Technical Report, University of Kansas (1994).
4. Brenan, K.E., Campbell, S.L., and Petzold, L.R., *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations* (Elsevier 1989).
5. Chun, H. M., Padilla, C. E, Chin, D. N., Watanabe, M., Karlov, V. I., Alper, H. E., Soosaar, K., Blair, K., Becker, O., L. Caves, S. D., Nagle, R., Haney, D. N., and Farmer, B. L. *J. Comp. Chem.*, 21, 159 (2000).
6. Dekker, K., and Verwer, J. G., *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, CWI Monographs (1984).
7. Dexuan Xie, "A New Parallel SOR Method", *SIAM J. Sci. Stat. Comput.*, submitted.
8. Dichmann D.J., Maddocks J.H., and Pego R.L., "Hamiltonian dynamics of an elastica and the stability of solitary waves", *Archive for Ratnl. Mech. and Anal.*, to appear 1996.
9. Ferrario, M., and Ryckaert, J.P., *Molec. Phys.*, **54** (3) 587, (1985).
10. Fratini A.V., Kopka M.L., Drew H.R., Dickerson R.E., *J. Mol. Biol.*, **163**, 129 (1983).
11. Haug, E.J., *Computer-Aided Kinematics and Dynamics of Mechanical Systems*, Allyn and Bacon, (1989).
12. Haug, E.J., and Deyo, R.C. (eds.), *Real-Time Integration Methods for Mechanical System Simulation*, NATO ASI Series **F 69** (Springer-Verlag, 1990).
13. Horiuchi T. and Go. N., *Proteins*, **10**, 106 (1991).
14. Ichiye T., Karplu M., *Proteins*, **11**, pp. 205-217 (1991).
15. Jain A., Viadehi N., and Rodriguez G., *J. Comput. Phya.*, **106**, pp. 258-268 (1993).
16. Leimkuhler, B.J., and Skeel, R.D., *J. Comp. Phys.*, **112**, 1, p. 117, (1994).
17. Leimkuhler B.J. and Skeel R.D., "Symplectic Numerical Integrators in Constrained Hamiltonian Systems", *J. Comp. Physics,* **112**, p. 117 (1994).
18. Brooks, B.R.; Bruccoleri, R.E.; Olafson, B.D.; States, D.J.; Swaminathan, S. and Karplus, M. *J. Comp Chem.*, **Vol. 4, No. 2**, 187-217, (1983).
19. Maddocks J.M. and Dichmann D.J., "Conservation laws in the dynamics of rods", *J. Elasticity*, **34**, pp. 83-96, (1994).
20. Maddocks J.M. and Dichmann D.J.,"An Impetus-Striction Simulation of the Dynamics of an Elastica", *J. Nonlinear Science*, **6**, pp. 271-292 (1996).
21. Maddocks J.M., Dichmann D.J., and Li Y., "Hamiltonian formulations and symmetries in rod mechanics", in *Mathematical Approaches to Biomolecular Structure and Dynamics*, The IMA Volumes in Mathematics and Its Applications, eds. Mesirov J.P. , Schulten K., & Sumners D.W., **82**, pp. 71-113 (1996) .

22. Maddocks J.M. , Dichmann D.J. and Pego R.L, "Hamiltonian dynamics of an elastica and the stability of solitary waves", *Arch. Rat. Mech. Anal.* **135**, pp. 357-396, (1996).

23. Maddocks J.M. and Kehrbaum S., "Effective properties of elastic rods with high intrinsic twist", submitted to *Physical Review E.*

24. Maddocks J.M. and Li, Y. ``On the Computation of Equilibria of Elastic Rods, Part I: Integrals, Symmetry and a Hamiltonian Formulation", *J. Comp. Physics*, to appear (63 pages)

25. Maddocks J.M. and Manning R.S. and Kahn J.D., "A Continuum Model of Sequence-Dependent DNA Structure", *J. Chemical Physics*, **105** No. 13, pp. 5626-5646 (1996).

26. Maddocks J.M. and Manning R.S., Paffenroth R.C. , Rogers K.A., and Warner, J.A. "Interactive computation, parameter continuation, and visualization", *Int. J. Bifurcation and Chaos*, to appear

27. Maddocks J.M and Overton M.L, "Stability Theory for Dissipatively Perturbed Hamiltonian Systems", *Comm. Pure and Applied Math.* **XLVIII**, pp. 583-610 (1995).

28. Maddocks J.M. and Pego R.L. , "An unconstrained Hamiltonian formulation for incompressible fluid flow," *Comm. Math. Physics*, **170**, pp. 207-217 (1995).

29. Maddocks J.M. and Xu J.-M., "Conservative Integrators in Hamiltonian Systems", preprint.

30. Mizuguchi K., Kidera A., and Go N., *Proteins*, **18**, 34 (1994).

31. Padilla, Carlos E. and Karlov, Valeri I., "Optimal Integrators for Reduced Variables Molecular Dynamics," Final Technical Report, *AFOSR-Sponsored Phase I SBIR*, Contract No. F49620-96-C-0035, July, 1997.

32. Roberson R.E., Schwertassek R., *Dynamics of Multibody Systems* (Springer-Verlag, 1988).

33. Reich S., "Numerical Integration of Highly Oscillatory Hamiltonian Systems Using Slow Manifolds", preprint (1994).

34. Ryckaert J.P., Ciccoti G., and Berendsen H.J.C., " Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of N-Alkanes", *J. of Comput. Physics*, **23**, 327-341 (1977).

35. Sanz-Serna J. M. and Calvo M. P. , *Numerical Hamiltonian Problems*, (London, Glasgow, New York, Tokyo, Melbourne, Madras: Chapman & Hall 1994).

36. Schiehlen, W. (ed.), *Advanced Multibody System Dyn.* (Kluwer Academic Publishers, 1993).

37. Schlick T., *Curr. Opin. Struct. Biol.*, **5**, pp. 245-262 (1995).

38. Schlick T., and Olson W.K., *J. Mol. Biol.*, **67**, pp. 2146-2166 (1994).

39. Scott L.R. and Xie D., "Parallel Linear Stationary Iterative Methods by Domain Decomposition", *SIAM J. Sci. Stat. Comput.*, submitted.

40. Shabana, A. A., *Dynamics of Multibody Systems* (John Wiley & Sons 1989).

41. Space B., Rabitz H., and Askar A., *J. Chem. Phys*, **99**, 9070 (1993).

42. Simo J.C. , Tarnow N., and Wong K.K. , "Exact Energy-Momentum Conserving Algorithms and Symplectic Schemes for Nonlinear Dynamics", *Computer Methods in Applied Mechanics and Engineering*, **1**, pp. 63-116, (1992).

43. Stuart A.M. and Humphries A.R., "Model Problems in Numerical Stability Theory for Initial Value Problems", *SIAM Review*, **36**, 2, pp. 226-257 (1994).

44. Swaminathan S., Harte W.E., and Beveridge D.L., *J. Am. Chem. Soc.*, **113**, 2717 (1991).

45. Tuckerman M., Berne B.J., and Martyna G.J., *J. Chem. Phys.*, **97**, 3, p. 1990-2001 (1992).

46. Turner J.D., Weiner P.K., and Wilkinson A.J., eds., ESCOM, Leiden (1993).

47. van Aalten D.M.F., Amadei A., Linssen A.B.M., Eijsink V.G.H., Vriend G., and Berendsen H.J.C., *Proteins*, **22**, pp. 45-54 (1995).

48. Verlet L., "Computer `experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones Molecules", *Phys. Rev.* **159**, pp. 98-103 (1967).

49. Reich S., "Symplectic Integrators for Systems of Rigid Bodies", Preprint (1995).

50. MATLAB User's Guide, The Mathwork's, Inc., Natick, Massachusetts, 1994.

51. Leimkulher, B.J., and Reich, S., Integration methods for molecular dynamics, *1994 IMA Summer Program: Molecular Biology*, Springer, 1994.

52. Schiehlen, W., *Advanced Multibody System Dynamics*, Kluwer Academic Publishers, 1993.

53. Allen, M.P., and Tildesley, D.J., *Computer Simulation of Liquids*, Oxford Science Publications, 1987.

54. Suzuki, M., *J. Math. Phys.*, **32**, 400, 1991.

55. Lambert, J., and Watson, D., *J. Inst., Math. Appllic.*, **18**, 189-202, 1976.

56. Simo, J., Karnow, N., Wong, K.K., *Comp. Meth. Appl., Mech. Eng*, **1**, 1528-44, 1992.

57. Ding, H.-Q., Karasawa, N., and Goddard III, W.A., *J. Chem. Phys.*, **97**, 4309, 1992.

58. Barth, E., and Leimkuhler, B., Symplectic Methods for Conservative Multibody Systems, *Workshop on Symplectic Integration Algorithms*, Fields Institute, 1993.

59. Rice, L.M., and Brünger, A.T., *Proteins: Structure, Function, and Genetics*, **19**, 277-290, 1994.

60. Chun, H.M., Turner, J.D., and Frisch, Paper AAS 89-457. *AAS/AIAA Astrodynamics Specialist Conference* (1989).

61. Turner, J.D., Weiner, P.S., Chun, H.M., Lupi, V., Gallion, S., and Singh, V.C., Variable Reduction Techniques Applied to Molecular Dynamics Simulations, *Computer Simulation of Biomolecular Systems Theoretical and Experimental Applications*, 2 Ch 24, Eds. W.F. Van Gunsteren, P.K. Weiner, A.J. Wilkenson, Escom, Leiden (1993).

62. Lapidus, L., Seinfeld, J.H., *Numerical Solution of Ordinary Differential Equations*, Academic Press, NY, 1971.

63. Gear, W.C., *Numerical Initial Value Problems in Ordinary Differential Equations*, Pretice-Hall, Inc., 1971.

64. Enhanced Molecular Dynamics Simulation Technology for Biotechnology Applications, *A Proposal in Response to the Advanced Technology Program (ATP)*, Submitted by Moldyn, Inc. to National Institute of Standards and Technology, 1994.

65. Enhanced Molecular Dynamics Simulation Technology for Biotechnology Applications, *ATP NIST project, Quarterly Report*, July-September 1996, Moldyn, Inc.

66. Bodley, C.S., Devers, A.D., Park, A.C., and Frish, H.P., A Digital Computer Program for Dynamic Interaction Simulation of Controls and Structure (DISCOS), *Vol. I, NASA Technical Paper 1219, Report No. G 7702-F26*, 1978.

67. Frish, H.P., Chun, H.M., and Turner, J.D., NDISCOS- Users and Programmers Manual, Photon Research Associates, Inc., Cambridge Division, 1993.

68. Watanabe, M., and Karplus, M., *J. Phys. Chem.*, **99**, 5680, 1995.

69. Schaefer, M. and Karplus M.A, Comprehensive Analytical Treatment of Continuum Electrostatic. *J. Phys. Chem.*, **100**, 1578-1599, 1996.